Optimization in High Dimensions via Accelerated, Parallel, and Proximal Coordinate Descent

SIAM Review
Vol. 58, No. 4, pp. xxx–xxx

Optimization in High Dimensions via Accelerated, Parallel, and Proximal Coordinate Descent*

Olivier Fercoq†
Peter Richtárik‡

Abstract. We propose a new randomized coordinate descent method for minimizing the sum of convex functions, each of which depends on a small number of coordinates only. Our method (APPROX) is simultaneously Accelerated, Parallel, and PROXimal; this is the first time such a method has been proposed. In the special case when the number of processors is equal to the number of coordinates, the method converges at the rate $2\bar{\omega}\bar{L}R^2/(k+1)^2$, where k is the iteration counter, $\bar{\omega}$ is a data-weighted *average* degree of separability of the loss function, $\bar{L}$ is the *average* of Lipschitz constants associated with the coordinates and individual functions in the sum, and R is the distance of the initial point from the minimizer. We show that the method can be implemented without the need to perform full-dimensional vector operations, which is the major bottleneck of accelerated coordinate descent, rendering it impractical. The fact that the method depends on the average degree of separability, and not on the maximum degree, can be attributed to the use of new safe large stepsizes, leading to improved expected separable overapproximation (ESO). These are of independent interest and can be utilized in all existing parallel randomized coordinate descent algorithms based on the concept of ESO. In special cases, our method recovers several classical and recent algorithms such as simple and accelerated proximal gradient descent, as well as serial, parallel, and distributed versions of randomized block coordinate descent. Due to this flexibility, APPROX had been used successfully by the authors in a graduate class setting as a modern introduction to deterministic and randomized proximal gradient methods. Our bounds match or improve upon the best known bounds for each of the methods APPROX specializes to. Our method has applications in a number of areas, including machine learning, submodular optimization, and linear and semidefinite programming.

Key words. randomized coordinate descent, acceleration, parallel methods, proximal methods, complexity, partial separability, convex optimization, big data

AMS subject classifications. 65K05, 90C25, 49M27, 68Q25, 68W10, 68W20, 65Y20

DOI. 10.1137/16M1085905

1. Introduction.

Developments in computing technology and the ubiquity of digital devices have resulted in an increased interest in solving optimization problems of

*Published electronically November 3, 2016. This paper originally appeared in *SIAM Journal on Optimization*, Volume 25, Number 4, 2015, pages 1997–2023. This work was supported by EPSRC grant EP/I017127/1 (Mathematics for Vast Digital Resources) and by the Centre for Numerical Algorithms and Intelligent Software (funded by EPSRC grant EP/G036136/1 and the Scottish Funding Council).

http://www.siam.org/journals/sirev/58-4/M108590.html

†LTCI, Télécom ParisTech, Institut Mines-Télécom, 75013 Paris, France (olivier.fercoq@ed.ac.uk).

‡School of Mathematics, University of Edinburgh, Edinburgh, United Kingdom (peter.richtarik@ed.ac.uk) The work of this author was supported by EPSRC grant EP/K02325X/1 (Accelerated Coordinate Descent Methods for Big Data Problems) and by the Simons Institute for the Theory of Computing at UC Berkeley.

extremely big sizes. Applications can be found in all areas of human endeavor where data is available, including the Internet, machine learning, data science, and scientific computing. The size of these problems is so large that it is necessary to decompose the problem into smaller, more manageable pieces. Traditional approaches, where it is possible to rely on full-vector operations in the design of an iterative scheme, must be revisited. Coordinate descent methods [23, 32] appear as a very popular class of algorithms for such problems as they can break down the problem into smaller pieces and can take advantage of sparsity patterns in the data. With big data problems it is necessary to design algorithms capable of utilizing modern parallel computing architectures. This resulted in an interest in parallel [33, 44, 18, 11, 29, 19, 26] and distributed [30, 17] coordinate descent methods.

In this work we focus on the solution of convex optimization problems with a huge number of variables of the form

$$\min_{x\in\mathbf{R}^N} \quad f(x)+\psi(x). \tag{1}$$

Here $x=(x^{(1)},\dots,x^{(n)})\in\mathbf{R}^N$ is a decision vector composed of n blocks with $x^{(i)}\in\mathbf{R}^{N_i}$, and $N=\sum_i N_i$. We assume that $f:\mathbf{R}^N\to\mathbf{R}$ is of the form

$$f(x)=\sum_{j=1}^{m} f_j(x), \tag{2}$$

where f_j are smooth convex functions, with $\psi:\mathbf{R}^N\to\mathbf{R}\cup\{+\infty\}$ being a convex (and lower semicontinuous) block separable regularizer (e.g., the $L1$ norm).

We now summarize the main contributions of this work.

1.1. Combination of Good Features. We design and analyze the first randomized block coordinate descent method (APPROX) which is simultaneously *accelerated*, *parallel*, and *proximal*. In fact, we are not aware of any published results on accelerated coordinate descent which would be either proximal *or* parallel. Our method is *accelerated* in the sense that it achieves an $O(1/k^2)$ convergence rate, where k is the iteration counter. The first *gradient* method with this convergence rate is due to Nesterov [21]; see also [47, 3]. An accelerated randomized coordinate descent method, for convex minimization without constraints, was originally proposed in 2010 by Nesterov [23].

Several variants of proximal and parallel (but nonaccelerated) randomized coordinate descent methods have been proposed [5, 33, 11, 30]. In Table 1 we provide a list[1] of some recent research papers proposing and analyzing *randomized* coordinate descent methods. The table substantiates our observation that while the block ("Blck" column) and proximal ("Prx" column) setup is relatively common in the literature, parallel methods ("Par" column) are much less studied, and there is just a handful of papers dealing with accelerated variants ("Acc" column). Moreover, existing accelerated methods are not efficient ("Eff" column)—with the exception of [13]—a point of crucial importance we will discuss next.

1.2. Efficient Iterations. We identify a large subclass of problems of the form (1) for which the *full-vector operations* inherent in accelerated methods *can be eliminated.* This contrasts with Nesterov's accelerated coordinate descent scheme [23], which is

[1] This list is necessarily incomplete; it was not our goal to be comprehensive. For a somewhat more substantial review of these and other works, we refer the reader to [33, 11].

Table 1 *An overview of selected recent papers proposing and analyzing the iteration complexity of* randomized *coordinate descent methods. The years correspond to the time the papers were first posted online (e.g., onto arXiv), and not the eventual publication time. "Eff" = the cost of each iteration is low (in particular, independent of the problem dimension N); "Blck" = works with blocks of coordinates; "Prx" = can handle proximal setup (has ψ term); "Par" = can update more blocks per iteration; "Acc" = accelerated, i.e., achieving the optimal $O(1/k^2)$ rate for nonstrongly convex objectives. Our algorithm has all these desirable properties. In the last column we highlight a single notable feature, necessarily chosen subjectively, of each work.*

Paper	Eff	Blck	Prx	Par	Acc	Notable feature
Leventhal and Lewis '08 [14]	✓	×	×	×	×	quadratic f
Shalev-Shwartz and Tewari '09 [38]	✓	×	ℓ_1	×	×	1st ℓ_1-regularized
Nesterov '10 [23]	×	✓	×	×	✓	1st blck and 1st acc
Richtárik and Takáč '11 [32]	✓	✓	✓	×	×	1st proximal
Bradley et al. '12 [5]	✓	×	ℓ_1	✓	×	ℓ_1-regularized parallel
Richtárik and Takáč '12 [33]	✓	✓	✓	✓	×	1st general parallel
Shalev-Shwartz and Zhang '12 [39]	✓	✓	✓	×	×	1st primal-dual
Necoara et al. '12 [20]	✓	✓	×	×	×	2-coordinate descent
Takáč et al. '13 [44]	✓	×	×	✓	×	1st primal-d. and parallel
Tappenden et al. '13 [45]	✓	✓	✓	×	×	1st inexact
Necoara and Clipici '13 [18]	✓	✓	✓	×	×	coupled constraints
Xiao and Lin '13 [51]	×	✓	×	×	✓	improvements on [23, 32]
Fercoq and Richtárik '13 [11]	✓	✓	✓	✓	×	1st nonsmooth f
Lee and Sidford '13 [13]	✓	×	×	×	✓	1st efficient accelerated
Richtárik and Takáč '13 [30]	✓	×	✓	✓	×	1st distributed
Liu et al. '13 [16]	✓	×	×	✓	×	1st asynchronous
Shalev-Shwartz and Zhang '13 [41]	✓	×	✓	×	✓	acceleration in the primal
Richtárik and Takáč '13 [29]	✓	×	×	✓	×	1st arbitrary sampling
This paper '13	✓	✓	✓	✓	✓	5 times ✓

impractical due to this bottleneck. Having established his convergence result, Nesterov made the following observation [23]:

> However, for some applications ... the complexity of one iteration of the accelerated scheme is rather high since for computing y_k it needs to operate with full-dimensional vectors.

Subsequently, in part due to these issues, the work of the community focused on simple methods as opposed to accelerated variants. For instance, Richtárik and Takáč [32] use Nesterov's observation to justify their focus on nonaccelerated methods in their work on coordinate descent methods in the proximal/composite setting.

Recently by a careful modification of Nesterov's method Lee and Sidford [13] were able to avoid full-dimensional operations in the case of minimizing a convex quadratic without constraints. This was achieved by introducing an extra sequence of iterates and observing that for quadratic functions it is possible to compute partial derivative of f evaluated at a linear combination of full-dimensional vectors without ever forming the combination. We extend the ideas of Lee and Sidford [13] to our general setting (1) in the case when $f_j(x) = \phi_j(a_j^T x)$, where ϕ_j are scalar convex functions with Lipschitz derivative and the vectors a_j are block-sparse.

1.3. Flexibility. APPROX is a remarkably versatile method, encoding several classical, recently developed, and new optimization methods as special cases. These variants are achieved by combinations of four design elements (see Table 2). In particular, by choosing to group all coordinates into a single block ($n = 1$), the only sensible

Table 2 *The methods in this table all arise as special cases of APPROX by varying four elements: the presence and form of the proximal term ψ in the problem formulation ("Prx"), the number of blocks n we decide to split the variable $x \in \mathbf{R}^N$ into ("Blck"), the choice of the block samplings $\hat{S}$, and the choice of the stepsize parameter θ_k. (GD = gradient descent; BCD = block coordinate descent.)*

Method	Prx ψ	Blck n	Sampling $\hat{S}$	θ_k
GD	0	1	$\hat{S} = \{1\}$ wp 1	constant
Projected GD	set indicator	1	$\hat{S} = \{1\}$ wp 1	constant
Proximal GD	any	1	$\hat{S} = \{1\}$ wp 1	constant
Acc Proximal GD [47, 3]	any	1	$\hat{S} = \{1\}$ wp 1	as in APPROX
Serial BCD [32]	separable	any	serial uniform	constant
Parallel BCD [33]	separable	any	any uniform	constant
Distributed BCD [30]	separable	any	distributed	constant
Acc Distr BCD [10]	separable	any	distributed	as in APPROX

sampling is to pick this block with probability 1, which makes the method deterministic. This corresponds to the first four methods in Table 2. To obtain the first three, we need to modify the stepsizes in APPROX (Algorithm 2) so that $\theta_k = \theta_0$ for all k. Doing this, we obtain simple (i.e., nonaccelerated) gradient descent in three varieties, depending on the choice of the proximal term: gradient descent (GD, no proximal term), projected GD (indicator function of a convex constraint set), and proximal GD. If we decrease the stepsizes as prescribed by APPROX, we recover Tseng's accelerated proximal gradient method. Let us now look at the last four methods in the table, all of which correspond to a setting with a nontrivial block decomposition and a general (but block-separable) proximal term. If we set $\theta_k = \theta_0$ for all k, we recover existing (nonaccelerated) serial (UCDC [32]), parallel (PCDM [33]), and distributed (Hydra [30]) coordinate descent methods, depending on the choice of the sampling. Finally, a follow-up paper to our work looks at APPROX specialized to a distributed sampling (Hydra^2 [10]). This last method was applied to solving a problem involving 50 billion variables.

1.4. New Stepsizes. We propose *new stepsizes* for parallel coordinate descent methods, based on a new expected separable overapproximation (ESO). These stepsizes can for some classes of problems (e.g., f_j = quadratics) be much larger than the stepsizes proposed for the (nonaccelerated) parallel coordinate descent method (PCDM) in [33]. Let ω_j be the number of blocks function f_j depends on. The stepsizes, and hence the resulting complexity, of PCDM depend on the quantity $\omega = \max_j \omega_j$. However, our stepsizes take all the values ω_j into consideration, and the result of this is a complexity that depends on a data-weighted average $\bar{\omega}$ of the values ω_j. Since $\bar{\omega}$ can be much smaller than ω, our stepsizes result in dramatic acceleration for our method and other methods whose analysis is based on an ESO [33, 11, 30].

1.5. Contents. The rest of the paper is organized as follows. In section 2 we outline a number of notable applications our method has found since it was first published [12]. We then describe new long stepsizes for parallel coordinate descent methods (applying more widely than to APPROX), based on novel assumptions, and compare them with existing stepsizes (section 3). The APPROX algorithm and the main complexity result are described in section 4. Subsequently, we give a proof of the result (section 5). We then describe an efficient implementation of our method, one that does not require the computation of full-vector operations (section 6), and finally comment on our numerical experiments (section 7).

1.6. Notation. It will be convenient to define natural operators acting between the spaces $\mathbf{R}^N$ and $\mathbf{R}^{N_i}$. In particular, we will often wish to lift a block $x^{(i)}$ from $\mathbf{R}^{N_i}$ to $\mathbf{R}^N$, filling the coordinates corresponding to the remaining blocks with zeros. Likewise, we will project $x \in \mathbf{R}^N$ back into $\mathbf{R}^{N_i}$. We will now formalize these operations.

Let U be the $N \times N$ identity matrix, and let $U = [U_1, U_2, \ldots, U_n]$ be its decomposition into column submatrices $U_i \in \mathbf{R}^{N \times N_i}$. For $x \in \mathbf{R}^N$, let $x^{(i)}$ be the block of variables corresponding to the columns of U_i, that is, $x^{(i)} = U_i^T x \in \mathbf{R}^{N_i}$, $i = 1, 2, \ldots, n$. Any vector $x \in \mathbf{R}^N$ can be written, uniquely, as $x = \sum_{i=1}^n U_i x^{(i)}$. For $h \in \mathbf{R}^N$ and $\emptyset \neq S \subseteq [n] \stackrel{\text{def}}{=} \{1, 2, \ldots, n\}$, we write

$$h_{[S]} = \sum_{i \in S} U_i h^{(i)}. \tag{3}$$

In other words, $h_{[S]}$ is a vector in $\mathbf{R}^N$ obtained from $h \in \mathbf{R}^N$ by zeroing out the blocks that do not belong to S. For convenience, we will also write

$$\nabla_i f(x) \stackrel{\text{def}}{=} (\nabla f(x))^{(i)} = U_i^T \nabla f(x) \in \mathbf{R}^{N_i} \tag{4}$$

for the vector of partial derivatives with respect to coordinates belonging to block i.

With each block $i \in [n]$ we associate a positive definite matrix $B_i \in \mathbf{R}^{N_i \times N_i}$ and a scalar $v_i > 0$, and equip $\mathbf{R}^{N_i}$ and $\mathbf{R}^N$ with the norms

$$\|x^{(i)}\|_{(i)} \stackrel{\text{def}}{=} \langle B_i x^{(i)}, x^{(i)} \rangle^{1/2}, \qquad \|x\|_v \stackrel{\text{def}}{=} \left(\sum_{i=1}^n v_i \|x^{(i)}\|_{(i)}^2 \right)^{1/2}. \tag{5}$$

The corresponding conjugate norms (defined by $\|s\|^* = \max\{\langle s, x \rangle : \|x\| \leq 1\}$) are

$$\|x^{(i)}\|_{(i)}^* \stackrel{\text{def}}{=} \langle B_i^{-1} x^{(i)}, x^{(i)} \rangle^{1/2}, \qquad \|x\|_v^* = \left(\sum_{i=1}^n v_i^{-1} (\|x^{(i)}\|_{(i)}^*)^2 \right)^{1/2}. \tag{6}$$

We also write $\|v\|_1 = \sum_i |v_i|$.

EXAMPLE 1 (blocks). *We now illustrate the above notation in two extreme situations:*

1. Blocks correspond to coordinates. *That is, $n = N$, and hence $N_i = 1$ for all i. In this case, $U_i = e_i$ is the ith unit coordinate vector, and hence $x^{(i)} = e_i^T x$ is the ith coordinate of x. For $h \in \mathbf{R}^N$, the vector $h_{[S]} \in \mathbf{R}^N$ has ith coordinate equal to $h^{(i)}$ if $i \in S$ and to 0 otherwise. The vector $\nabla_i f(x) = (\nabla f(x))^{(i)} = e_i^T \nabla f(x)$ is the ith partial derivative of f at x. Primal block norm $\|x^{(i)}\|_{(i)}$ reduces to $B_i^{1/2}|x^{(i)}|$ for some positive scalar B_i, and the primal norm in $\mathbf{R}^N$ is a weighted Euclidean norm: $\|x\|_v = (\sum_{i=1}^n v_i (x^{(i)})^2)^{1/2}$. The dual norms have an analogous meaning.*
2. All coordinates belong to a single block. *That is, $n = 1$, and hence $N_1 = N$. In this case, $U_1 = I$ is the identity matrix, and hence $x^{(1)} = x$. Further, $h_{[S]} = h$ if $S = \{1\}$ and $h_{[S]} = 0$ if $S = \emptyset$. The vector $\nabla_1 f(x)$ is the gradient of f at x. The primal block norm $\|x^{(1)}\|_{(1)}$ is simply equal to $\langle B_1 x, x \rangle^{1/2}$, and the primal norm in $\mathbf{R}^N$ is a weighted version thereof: $\|x\|_v = \sqrt{v_1} \langle B_1 x, x \rangle^{1/2}$.*

Table 3 *Selected applications of APPROX, developed by others after the publication of* [12].

Application	Paper	Section
Empirical risk minimization	[10, 15]	2.1
Submodular optimization	Ene and Nguyen [9]	2.2
Packing and covering linear programs	Allen-Zhu and Orecchia [1]	2.3
Least-squares semidefinite programming	Sun, Toh, and Yang [43]	2.4

2. Applications. In this section we describe four applications areas for the APPROX algorithm, all motivated and building on the developments in the earlier version of this paper where the APPROX method was first developed [12] (see Table 3). This section therefore contains new material not present in [12]. It is not necessary for the reader at this point to know any details about APPROX beyond what was mentioned in the introduction; the purpose here is to stress that the method has a wide array of applications.

2.1. Empirical Risk Minimization. Empirical risk minimization (ERM) is a powerful and immensely popular paradigm for training statistical (machine) learning models [37]. In statistical learning, one wishes to "learn" an unknown function $h^* : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ (set of samples) and $\mathcal{Y}$ (set of labels) are arbitrary domains. Roughly speaking, the goal of statistical learning is to find a function (predictor, hypothesis) $h : \mathcal{X} \to \mathcal{Y}$ from some predefined set (hypothesis class) $\mathcal{H}$ of predictors which in some statistical sense is the best approximation of h^*. In particular, we assume that there is an *unknown* distribution $\mathcal{D}$ over $\xi \in \mathcal{X}$. Given a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbf{R}$, we define the *risk* (generalization error) associated with predictor $h \in \mathcal{H}$ to be

$$L_{\mathcal{D}}(h) = \mathbf{E}_{\xi \sim \mathcal{D}}\, \ell(h(\xi), h^*(\xi)). \tag{7}$$

The goal of statistical learning is to find $h \in \mathcal{H}$ of minimal risk:

$$\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h). \tag{8}$$

A natural, albeit in general intractable, choice of a loss function in some applications is $\ell(y, y') = 0$ if $y = y'$ and $\ell(y, y') = 1$ otherwise.

Let $\mathcal{X}$ be a collection of images, let $\mathcal{Y} = \{-1, 1\}$, and let $h^*(\xi)$ be 1 if image ξ contains an image of a cat, and $h^*(\xi) = -1$ otherwise. If we are able to learn h^*, we will be able to detect images of cats. Problems where $\mathcal{Y}$ consist of two elements are called *classification problems*. The domain set can instead represent a video collection, a text corporus, a collection of emails, or any other collection of objects which we can represent mathematically. If $\mathcal{Y}$ is a finite set consisting of more than two elements, we speak of *multiclass classification*. If $\mathcal{Y} = \mathbf{R}$, we speak of *regression*.

One of the fundamental issues making (8) difficult to solve is the fact that the distribution $\mathcal{D}$ is not known. ERM is a paradigm for overcoming this obstacle, assuming that we have access to independent samples from $\mathcal{D}$. In ERM, we first collect a *training set* of independent and identically distributed samples and their labels; that is, $\mathcal{S} = \{(\xi_j, y_j) \in \mathcal{X} \times \mathcal{Y} \ : \ j = 1, 2, \ldots, m\}$, where $y_j = h^*(\xi_j)$. Subsequently, we replace the expectation in (7) defining the risk by a sample average approximation, which defines the *empirical risk*:

$$L_{\mathcal{S}}(h) = \frac{1}{m} \sum_{j=1}^{m} \ell(h(\xi_j), y_j).$$

The ERM paradigm is to solve the *empirical risk minimization* problem

$$\min_{h\in\mathcal{H}} L_{\mathcal{S}}(h) \tag{9}$$

instead of the harder risk minimization problem (8). In practice, $\mathcal{H}$ is often chosen to be a parametric class of functions described by a parameter $x \in \mathbf{R}^d$. For instance, let $\mathcal{X} \subseteq \mathbf{R}^d$ (d = number of features) and $\mathcal{Y} = \mathbf{R}$, and consider the class of *linear predictors* $\mathcal{H} = \{h : h(\xi) = x^\top \xi\}$. Clearly, h is uniquely defined by $x \in \mathbf{R}^d$. Defining $\ell_j : \mathbf{R} \to \mathbf{R}$ via $\ell_j(t) = \frac{1}{m}\ell(t, y_j)$, and setting $f_j(x) = \ell_j(\xi_j^T x)$, we have

$$f(x) \stackrel{\text{def}}{=} \sum_{j=1}^m f_j(x) = \sum_{j=1}^m \ell_j(\xi_j^T x) = L_{\mathcal{S}}(h).$$

Hence, the ERM problem fits our framework (1)+(2), with $\psi \equiv 0$. However, in practice one often uses nonzero ψ, which is interpreted as a regularizer, and is included in order to prevent overfitting and hence allow the estimator to generalize to future, unobserved samples.

If the number of features is larger than the number of examples ($d \gg m$), randomized coordinate descent is an efficient algorithm for solving (1) [32, 33, 40]. If the training set $\mathcal{S}$ is so large that it does not fit the memory (or disk space) of a single machine, one needs to employ a distributed computing system and solve ERM via a distributed optimization algorithm. One option is the use of distributed coordinate descent [31, 17], known as Hydra. APPROX has been successfully applied in the distributed setting, leading to the Hydra2 method [10]. In this work, the authors solve an ERM problem involving a training set of several terabytes in size, and 50 billion features.

If the number of examples in the training set is larger than the number of features ($m \gg d$), it is typically not efficient to employ randomized coordinate descent, or APPROX, to the ERM problem directly. Instead, the state-of-the-art methods are variants of randomized coordinate descent applied to the *dual* problem.[2]

The (Fenchel) dual of the regularized ERM problem for linear predictors considered above has the form

$$\min_{y\in\mathbb{R}^m} \psi^*\left(\frac{1}{m}\sum_{j=1}^m y_j\xi_j\right) + \frac{1}{m}\sum_{j=1}^m \ell_j^*(-y_j),$$

where ψ^* (resp., ℓ_j^*) is the Fenchel conjugate of ψ (resp., ℓ_j). The function $y \mapsto \psi^*(\frac{1}{m}\sum_j y_j\xi_j)$ has Lipschitz gradient if we assume that ψ is strongly convex, and $y \mapsto \frac{1}{m}\sum_{j=1}^m \ell_j^*(-y_j)$ is separable. This also fits the framework (1)+(2), f corresponding to the first part of the objective (and consisting of a single summand), and ψ corresponding to the second part of the objective (block separability is implied by separability).

We developed APPROX with ERM as an application in mind, and hence our numerical experiments in section 7 consider two key ERM problems: the Lasso problem and the support vector machine (SVM) problem.

[2]For a detailed comparison of the application of coordinate descent to the primal vs. dual problem, we refer the reader to the work of Csiba and Richtárik [7]. One of the conclusions of that work is that in the case of dense training data and $L2$ regularizer, primal coordinate descent is better than dual coordinate descent, in theory, precisely when $d > m$. For sparse and structured data, the answer is more intricate.

Following our paper, Lin, Lu, and Xiao [15] proposed a version of APPROX designed for *strongly* convex problems. Their motivation was that practitioners often choose regularizers that are at the same time separable *and* strongly convex. This leads to problems for which we have a good lower bound on the strong convexity parameter. With this additional knowledge, they showed that the rate of convergence of a properly modified APPROX algorithm, applied to the dual problem, leads to state-of-the-art complexity for a class of ERM problems.

2.2. Submodular Optimization. Ene and Nguyen [9] showed how the APPROX algorithm leads to a state-of-the-art method for minimizing *decomposable submodular functions*. Submodular minimization has a vast and growing array of applications, including image segmentation [36, 9], graphical model structure learning, experimental design, Bayesian variable selection, and minimizing matroid rank functions [2].

We now briefly introduce the notion of submodularity. Let $V = \{1, 2, \dots, d\}$ be a finite ground set. A real-valued set function $\phi : 2^V \to \mathbf{R}$ is called *modular* if $\phi(\emptyset) = 0$ and there exists a vector $w \in \mathbf{R}^d$ such that $\phi(A) = \sum_{i \in A} w_i$ for all $\emptyset \neq A \subseteq V$. It is called *submodular* if $\phi(A)+\phi(B) \geq \phi(A \cap B)+\phi(A \cup B)$ for any two sets $A, B \subseteq V$. An equivalent and often more intuitive characterization of submodularity is the following "diminishing returns property": ϕ is submodular if and only if for all $A \subseteq B \subseteq V$ and $k \in V$ such that $k \notin B$ we have $\phi(A \cup \{k\}) - \phi(A) \geq \phi(B \cup \{k\}) - \phi(B)$.

Ene and Nguyen [9] consider the *decomposable* submodular minimization problem

$$\min_{A \subseteq V} \sum_{i=1}^{n} \phi_i(A), \tag{10}$$

where $\phi_i : 2^V \to \mathbf{R}$ are simple submodular functions (simplicity refers to the assumption that it is simple to minimize ϕ_i plus a modular function). Instead of solving (10) directly, one can focus on solving the unconstrained convex minimization problem

$$\min_{z \in \mathbf{R}^d} \sum_{i=1}^{n} \left(\hat{\phi}_i(z) + \frac{1}{2n} \|z\|^2 \right), \tag{11}$$

where $\|\cdot\|$ is the standard Euclidean norm, and $\hat{\phi}_i : \mathbf{R}^d \to \mathbf{R}$ is the Lovász extension of ϕ_i (i.e., the support function of the base polytope $P_i \subset \mathbf{R}^d$ of ϕ_i). Given a solution z, once recovers the solution of (10) by setting

$$A = A(z) = \{k \in V \ : \ z_k \geq 0\}. \tag{12}$$

Further, instead of solving (11), one focuses on its (Fenchel) dual:

$$\min_{x^{(1)} \in P_1, \dots, x^{(n)} \in P_n} f(x) \stackrel{\text{def}}{=} \frac{1}{2} \left\| \sum_{i=1}^{n} x^{(i)} \right\|^2. \tag{13}$$

It can be shown that if $x = (x^{(1)}, \dots, x^{(n)}) \in \mathbf{R}^{nd} \stackrel{\text{def}}{=} \mathbf{R}^N$ solves (13), then

$$z = -\sum_{i=1}^{n} x^{(i)} \tag{14}$$

solves (11). Note that f is a convex quadratic function. If we let ψ be the indicator function of the set $P \stackrel{\text{def}}{=} P_1 \times \cdots \times P_n \subseteq \mathbf{R}^N$, i.e., $\psi(x) = 0$ if $x \in P$ and $\psi(x) = +\infty$ otherwise, then (13) is of the form (1), where $N_i = d$ for all i. It remains to apply the APPROX method to this problem and transform the solution back via (14) and then (12) to obtain solution of the original problem (10).

2.3. Packing and Covering Linear Programs. Packing and covering problems are a pair of mutually dual linear programming problems of the form

$$\begin{aligned} \text{Packing LP:} \quad & \max_{x\geq 0}\{1^T x \ : \ Ax \leq 1\}, \\ \text{Covering LP:} \quad & \min_{y\geq 0}\{1^T y \ : \ A^T y \leq 1\}, \end{aligned}$$

where A is a real matrix, and 1 denotes the vector of appropriate dimension with all entries equal to 1. These problems become more difficult as the size of A grows since each iteration of interior-point solvers becomes more expensive.

Allen-Zhu and Orecchia [1] developed algorithms for these problems whose complexity is $O(N_A \log(N_A)\log(\epsilon^{-1})/\epsilon)$ for packing and $O(N_A \log(N_A)\log(\epsilon^{-1})\epsilon^{-1.5})$ for covering LP, respectively. This complexity is nearly linear in the size of the problem $N_A = \text{nnz}(A)$ (number of nonzero entries in A), does not depend on the magnitude of the elements of A, and has a better dependence on the accuracy ϵ than other nearly linear time approaches. The improvement in the complexity is due to the use of accelerated proximal coordinate descent techniques such as those developed in our paper, combined with extra techniques, such as the use of an exponential penalty.

2.4. Least Squares Semidefinite Programming. Semidefinite programming has a very important role in optimization due to its ability to model and efficiently solve a wide array of problems appearing in fields such as control, network science, signal processing, and computer science [48, 46, 4, 49]. In semidefinite programming, one aims to minimize a linear function in a matrix variable, subject to linear equality and inequality constraints and the additional requirement that the matrix variable be positive semidefinite.

Sun, Toh, and Yang [43] consider the canonical semidefinite program (SDP)

$$\begin{aligned} \min_{X\in\mathcal{S}^n_+,\ s\in\mathbf{R}^{m_I}} \quad & \langle C, X\rangle \\ \text{subject to} \quad & \mathcal{A}_E(X) = b_E,\ \mathcal{A}_I(X) = s,\ L \leq X \leq U,\ l \leq s \leq u, \end{aligned}$$

where $\langle C, X\rangle$ is the trace inner product, $\mathcal{S}^n_+$ is the cone of $n \times n$ symmetric positive semidefinite matrices, $\mathcal{A}_E : \mathcal{S}^n_+ \to \mathbf{R}^{m_E}$ and $\mathcal{A}_I : \mathcal{S}^n_+ \to \mathbf{R}^{m_I}$ are linear maps, $L \leq U$ are given positive semidefinite matrices, and $l \leq u$ are given vectors in $\mathbf{R}^{m_I}$.

The above SDP can be solved by a proximal point algorithm (PPA) of Rockafellar [34, 35]. In each iteration of PPA, one needs to solve a *least-squares semidefinite program (LS-SDP)* of the form

$$\begin{aligned} (X^{k+1}, s^{k+1}) = \arg \min_{X\in\mathcal{S}^n_+, s\in\mathbb{R}^{m_I}} \quad & \langle C, X\rangle + \frac{1}{2\sigma_k}(\|X - X^k\|^2 + \|s - s^k\|^2) \\ \text{subject to} \quad & \mathcal{A}_E(X) = b_E,\ \mathcal{A}_I(X) = s,\ L \leq X \leq U,\ l \leq s \leq u, \end{aligned}$$

where (X^k, s^k) is the previous iterate, and $\sigma_k > 0$ a regularization parameter. Sun, Toh, and Yang [43] observe that the dual of LS-SDP has block-separable constraints and can hence be written in the form (1), with either 2 or 4 blocks ($n = 2$ or $n = 4$). They used this observation as a starting point to propose new algorithms for LS-SDP that combine advanced linear algebra techniques, a fine study of the errors made by each inner solver, and coordinate descent ideas. They consider APPROX and block coordinate descent as natural competitors to their specialized methods. They implemented the methods using 4 blocks, each equipped with a nontrivial norm

defined by a well-chosen positive semidefinite matrix $B_i \in \mathbf{R}^{N_i \times N_i}$ and approximate solutions to the proximity operators. Finally, Sun, Toh, and Yang conducted extensive experiments on 616 SDP instances coming from relaxations of combinatorial problems. It is worth noting that on these instances, APPROX is vastly faster than standard (nonaccelerated) block coordinate descent.

3. Stepsizes for Parallel Coordinate Descent Methods. The framework for designing and analyzing (nonaccelerated) PCDMs, developed by Richtárik and Takáč [33], is based on the notions of *block sampling* and *expected separable overapproximation (ESO)*. We now briefly review this framework, as our accelerated method is cast in it, too. Informally, a block sampling is the random law describing the *selection of blocks* in each iteration. An ESO is an inequality, involving f and $\hat{S}$, defining stepsize/ESO constants $v_1, \ldots . v_n$, which are in turn used to *compute updates* to selected blocks. The complexity analysis in our paper is based on the following generic assumption.

ASSUMPTION 1 (expected separable overapproximation [33, 11]).

1. *f is convex and differentiable.*
2. *$\hat{S}$ is a uniform block sampling. That is, $\hat{S}$ is a random subset of $[n] = \{1, 2, \ldots, n\}$ with the property*[3] *that $\mathbf{P}(i \in \hat{S}) = \mathbf{P}(j \in \hat{S})$ for all $i, j \in [n]$. Let $\tau \overset{\text{def}}{=} \mathbf{E}[|\hat{S}|]$.*
3. *There are computable constants $v = (v_1, \ldots, v_n) > 0$ for which the pair $(f, \hat{S})$ admits the ESO*

$$\mathbf{E}\left[f(x + h_{[\hat{S}]})\right] \leq f(x) + \frac{\tau}{n}\left(\langle \nabla f(x), h\rangle + \frac{1}{2}\|h\|_v^2\right), \quad x, h \in \mathbf{R}^N. \tag{15}$$

If the above inequality holds, for brevity we will write[4] *$(f, \hat{S}) \sim \mathrm{ESO}(v)$, indicating that v depends both on f and $\hat{S}$.*

In the context of PCDMs, and for uniform samplings, the ESO inequality (15) was introduced and systematically studied by Richtárik and Takáč [33]. An ESO inequality for *distributed* sampling was developed in [30] and further refined in [10]. A PCDM with a nonuniform sampling, and the associated nonuniform ESO inequality, was proposed in [29].

Note that part 1 of the above assumption involves f only; that is, it is an assumption on the smooth part of the objective function describing the problem. On the other hand, part 2 of the assumption is fully in the hands of the practitioner—and is in principle independent of the problem itself. That is, $\hat{S}$ is a parameter of the method, and hence one can *decide* how to choose this parameter. Note that we restrict our attention to *uniform* samplings. However, the results of this paper were extended after the publication of [12], and, in particular, the uniformity assumption was lifted. This also necessitated a change in definition of the ESO inequality (15), and in the APPROX algorithm (the extended algorithm is referred to by the name ALPHA [26]). A detailed explanation of why (15) is a reasonable assumption is given

[3] It is easy to see that if $\hat{S}$ is a uniform sampling, then, necessarily, $\mathbf{P}(i \in \hat{S}) = \frac{\mathbf{E}[|\hat{S}|]}{n}$ for all $i \in [n]$.

[4] In [33], the authors write $\frac{\beta}{2}\|h\|_w^2$ instead of $\frac{1}{2}\|h\|_v^2$. This is because they study families of samplings $\hat{S}$, parameterized by τ, for which w is fixed and all changes are captured in the constant β. Clearly, the two definitions are interchangeable, as one can choose $v = \beta w$. Here we will need to compare weights which are not linearly dependent, hence the simplified notation.

in [33, 11]; here we shall only provide a brief commentary. Recall that the modeler can choose how the space $\mathbf{R}^N$ is decomposed into n blocks. If we choose $n = 1$, then all coordinates belong to a single block, all randomness is removed from (15), and APPROX specializes to one of the variants of gradient descent from Table 2. The ESO inequality then simply requires the gradient of f to be Lipschitz with constant v with respect to the norm $\|\cdot\|_{(1)}$ (compare this with Theorem 1(ii)). Inequality (15) is the natural extension of this to the case when it is only allowed to move in a random subspace of $\mathbf{R}^N$. Note that this assumption is *always satisfied* if the gradient of f is Lipschitz, for instance, for *large-enough* constants $\{v_i\}$. These constants determine the stepsizes in our method, and hence we need to have easy-to-compute formulas for $\{v_i\}$. It turns out that the complexity bound of APPROX improves as these constants get smaller, which means that tighter bounds are preferable.

Fercoq and Richtárik [11, Theorem 10] observed that inequality (15) is equivalent to requiring that the gradients of the functions

$$\hat{f}_x : h \mapsto \mathbf{E}\left[f(x + h_{[\hat{S}]})\right], \qquad x \in \mathbf{R}^N,$$

be Lipschitz at $h = 0$, uniformly in x, with constant τ/n, with respect to the norm $\|\cdot\|_v$. Equivalently, the Lipschitz constant is $L^{\hat{f}}$ with respect to the norm $\|\cdot\|_{\tilde{v}}$, where

$$L^{\hat{f}} = \tau\|v\|_1/n^2, \qquad \tilde{v} \stackrel{\text{def}}{=} nv/\|v\|_1.$$

The change of norms is done so as to enforce the weights in the norm to add up to n, which would roughly enable us to compare different ESOs via constants $L^{\hat{f}}$. The above observations are useful in understanding what the ESO inequality encodes: By moving from x to $x_+ = x + h_{[\hat{S}]}$, one is taking a step in a random subspace of $\mathbf{R}^N$ spanned by the blocks belonging to $\hat{S}$. If $\tau \ll n$, which is often the case in big data problems,[5] the step is confined to a *low-dimensional* subspace of $\mathbf{R}^N$. It turns out that for many classes of functions arising in applications, for instance, for functions exhibiting certain sparsity or partial separability patterns, it is the case that the gradient of f varies much more slowly in such subspaces, on average, than it does in $\mathbf{R}^N$. This in turn would imply that updates h based on minimizing the right-hand side of (15) would produce larger steps and eventually lead to faster convergence.

3.1. New Model. Consider f of the form (2), i.e., $f(x) = \sum_{j=1}^m f_j(x)$, and let $C_j \neq \emptyset$ be the set of blocks function f_j depends on. Define $\omega_j \stackrel{\text{def}}{=} |C_j|$, and $\omega \stackrel{\text{def}}{=} \max_j \omega_j$. Clearly, any function f is of this form: it suffices to choose $m = 1$ and $C_1 = \{1, 2, \ldots, n\}$. However, many functions appearing in applications, notably in machine learning and statistics, have a natural representation of this form with m being large and $\omega_j \ll n$ for some, most, or all j.

ASSUMPTION 2. *The functions $\{f_j\}$ have block-Lipschitz gradient with constants $L_{ji} \geq 0$. That is, for all $j = 1, 2, \ldots, m$ and $i = 1, 2, \ldots, n$,*

$$\|\nabla_i f_j(x + U_i t) - \nabla_i f_j(x)\|_{(i)}^* \leq L_{ji}\|t\|_{(i)}, \qquad x \in \mathbf{R}^N,\ t \in \mathbf{R}^{N_i}. \tag{16}$$

Note that, under the above assumption, we necessarily have

$$L_{ji} = 0 \qquad \text{whenever} \qquad i \notin C_j. \tag{17}$$

[5] In fact, one may define a "big data" problem by requiring that the number of parallel processors τ available for optimization be much smaller than the dimension n of the problem.

Assumption 2 is *stronger* than the assumption considered in [33]. Indeed, in [33] the authors only assumed that the *sum*, f, as opposed to the individual functions f_j, has a block-Lipschitz gradient, with constants $L_1, \dots, L_n$:

$$\|\nabla_i f(x + U_i t) - \nabla_i f(x)\|^*_{(i)} \leq L_i \|t\|_{(i)}, \qquad x \in \mathbf{R}^N,\ t \in \mathbf{R}^{N_i}.$$

It is easy to see that if the stronger condition is satisfied, then the weaker one is also satisfied with $L_i \leq \sum_{j=1}^m L_{ji}$.

3.2. New ESO. The main result of this section is Theorem 1, in which we derive an ESO inequality for functions satisfying Assumption 2 and the τ-nice sampling (for some $\tau \in [n]$). A sampling is called τ-nice if it picks a set of size τ uniformly at random from all subsets of size τ. It is, however, possible to derive similar bounds for *all uniform samplings* considered in [33] using the same approach. For an alternative approach to deriving ESO inequalities, one relying on a different assumption on f and capable of handling arbitrary samplings, we refer the reader to [27].[6]

In the proof we will refer to two identities established in [33]. First, for the τ-nice sampling and any set $J \subseteq [n]$, we have the identity

$$\mathbf{E}[|J \cap \hat{S}|^2] = \frac{|J|\tau}{n}\left(1 + \frac{(|J|-1)(\tau-1)}{\max\{1, n-1\}}\right). \tag{18}$$

If, moreover, $\theta_1, \dots, \theta_n$ are arbitrary real scalars and $\mathbf{P}(|J \cap \hat{S}| = k) > 0$, then

$$\mathbf{E}\left[\sum_{i \in J \cap \hat{S}} \theta_i \mid |J \cap \hat{S}| = k\right] = \frac{k}{|J|}\sum_{i \in J} \theta_i. \tag{19}$$

We are now ready to state and prove our result.

THEOREM 1. *Let f satisfy Assumption 2.*

(i) *If $\hat{S}$ is a τ-nice sampling, then for all $x, h \in \mathbf{R}^N$,*

$$\mathbf{E}\left[f(x + h_{[\hat{S}]})\right] \leq f(x) + \frac{\tau}{n}\left(\langle \nabla f(x), h\rangle + \frac{1}{2}\|h\|_v^2\right), \tag{20}$$

where

$$v_i \stackrel{\text{def}}{=} \sum_{j=1}^m \beta_j L_{ji} = \sum_{j: i \in C_j} \beta_j L_{ji}, \qquad i = 1, 2, \dots, n, \tag{21}$$

$$\beta_j \stackrel{\text{def}}{=} 1 + \frac{(\omega_j - 1)(\tau - 1)}{\max\{1, n-1\}}, \qquad j = 1, 2, \dots, m.$$

That is, $(f, \hat{S}) \sim ESO(v)$.

(ii) *As a corollary to part* (i)*, for all $x, h \in \mathbf{R}^N$ we have*

$$f(x+h) \leq f(x) + \langle \nabla f(x), h\rangle + \frac{1}{2}\|h\|_{v'}^2 = f(x) + \langle \nabla f(x), h\rangle + \frac{\bar{\omega}\bar{L}}{2}\|h\|_w^2, \tag{22}$$

where $v'_i = \sum_j \omega_j L_{ji}$, and $\bar{\omega}$, $\bar{L}$, and $w = (w_1, \dots, w_n)$ are defined by

$$\bar{\omega} \stackrel{\text{def}}{=} \sum_{j=1}^m \omega_j \frac{\sum_i L_{ji}}{\sum_{k,i} L_{ki}}, \quad \bar{L} \stackrel{\text{def}}{=} \frac{\sum_{ji} L_{ji}}{n}, \quad w_i \stackrel{\text{def}}{=} \frac{n}{\sum_{j,i} \omega_j L_{ji}} \sum_{j=1}^m \omega_j L_{ji}. \tag{23}$$

[6] This paper appeared after [12].

Note that $v' = \bar{\omega}\bar{L}w$, $\sum w_i = n$, *and that* $\bar{\omega}$ *is a data-weighted* average *of the values* $\{\omega_j\}$.

Proof. Statement (ii) is a special case of (i) for $\tau = n$ (notice that for n-nice sampling we have $v = v'$ and $\bar{\omega}\bar{L}w = v$). We hence only need to prove (i). A well-known consequence of (16) is

$$f_j(x + U_i t) \le f_j(x) + \langle \nabla_i f_j(x), t\rangle + \frac{L_{ji}}{2}\|t\|^2_{(i)}, \qquad x \in \mathbf{R}^N,\ t \in \mathbf{R}^{N_i}. \tag{24}$$

We first claim that for all j,

$$\mathbf{E}\left[f_j(x + h_{[\hat{S}]})\right] \le f_j(x) + \frac{\tau}{n}\left(\langle \nabla f_j(x), h\rangle + \frac{\beta_j}{2}\|h\|^2_{L_{j:}}\right), \tag{25}$$

where $L_{j:} = (L_{j1}, \dots, L_{jn}) \in \mathbf{R}^n$. That is, $(f_j, \hat{S}) \sim ESO(\beta_j L_{j:})$. Inequality (20) then follows by adding up[7] the inequalities (25) for all j. Let us now prove the claim.[8] We fix x and define

$$\hat{f}_j(h) \stackrel{\text{def}}{=} f_j(x + h) - f_j(x) - \langle \nabla f_j(x), h\rangle. \tag{26}$$

Since $\mathbf{E}[\hat{f}_j(h_{[\hat{S}]})] \stackrel{(26)}{=} \mathbf{E}[f_j(x + h_{[\hat{S}]})] - f_j(x) - \frac{\tau}{n}\langle \nabla f_j(x), h\rangle$, it now only remains to show that

$$\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]})\right] \le \frac{\tau\beta_j}{2n}\|h\|^2_{L_{j:}}. \tag{27}$$

We adopt the convention that expectation conditional on an event which happens with probability 0 is equal to 0. Letting $\eta_j \stackrel{\text{def}}{=} |C_j \cap \hat{S}|$, we can now write

$$\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]})\right] = \sum_{k=0}^{n} \mathbf{P}(\eta_j = k)\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]}) \mid \eta_j = k\right]. \tag{28}$$

For any $k \ge 1$ for which $\mathbf{P}(\eta_j = k) > 0$, we now use convexity of $\hat{f}_j$ to write

$$\begin{aligned}
\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]}) \mid \eta_j = k\right] &= \mathbf{E}\left[\hat{f}_j\left(\frac{1}{k}\sum_{i \in C_j \cap \hat{S}} kU_i h^{(i)}\right) \mid \eta_j = k\right] \\
&\le \mathbf{E}\left[\frac{1}{k}\sum_{i \in C_j \cap \hat{S}} \hat{f}_j\left(kU_i h^{(i)}\right) \mid \eta_j = k\right] \\
&\stackrel{(19)}{=} \frac{1}{\omega_j}\sum_{i \in C_j} \hat{f}_j\left(kU_i h^{(i)}\right) \\
&\stackrel{(24)+(26)}{\le} \frac{1}{\omega_j}\sum_{i \in C_j} \frac{L_{ji}}{2}\|kh^{(i)}\|^2_{(i)} = \frac{k^2}{2\omega_j}\|h\|^2_{L_{j:}}.
\end{aligned} \tag{29}$$

[7] At this step we could have also simply applied Theorem 10 from [33], which gives the formula for an ESO for a conic combination of functions given ESOs for the individual functions. The proof, however, also amounts to simply adding up the inequalities.

[8] This claim is a special case of Theorem 14 in [33], which gives an ESO bound for a *sum* of functions f_j (here we only have a single function). We include the proof, since in this special case it is more straightforward.

Finally, combining (28) and (29), we get (27):

$$\mathbf{E}\left[\hat{f}_j(h_{[\hat{S}]})\right] \overset{(29)+(28)}{\leq} \sum_k \mathbf{P}(\eta_j = k)\frac{k^2}{2\omega_j}\|h\|^2_{L_{j:}} = \frac{1}{2\omega_j}\|h\|^2_{L_{j:}}\mathbf{E}[|C_j\cap\hat{S}|^2] \overset{(18)}{=} \frac{\tau\beta_j}{2n}\|h\|^2_{L_{j:}},$$

which concludes the proof. ∎

Note that (23) says that the gradient of f is Lipschitz with respect to the norm $\|\cdot\|_w$ with constant $\bar{\omega}\bar{L}$. We write (23) in terms of the norm $\|\cdot\|_w$ since the weights w_i add up to n, and hence the norm is in some sense comparable in scale to the standard Euclidean norm. The quantities β_j have a natural interpretation. It can be inferred from the identities established in [33, section 4] that $\beta_j = \mathbf{E}[|C_j \cap \hat{S}|^2]/\mathbf{E}[|C_j \cap \hat{S}|]$. Alternatively, it can be seen that β_j is the expected size of $|C_j \cap \hat{S}|$ conditioned on the event that the intersection is nonempty.

3.3. Computation of L_{ji}. We now give a formula for the constants L_{ji} in the case when f_j arises as a composition of a scalar function ϕ_j, whose derivative has a known Lipschitz constant (this is often easy to compute), and a linear functional. Let A be an $m \times N$ real matrix, and for $j \in \{1, 2, \ldots, m\}$ and $i \in [n]$ define

$$A_{ji} \overset{\text{def}}{=} e_j^T A U_i \in \mathbf{R}^{1\times N_i}. \tag{30}$$

That is, A_{ji} is a row vector composed of the elements of row j of A corresponding to block i.

THEOREM 2. *Let $f_j(x) = \phi_j(e_j^T Ax)$, where $\phi_j : \mathbf{R} \to \mathbf{R}$ is a function with L_{ϕ_j}-Lipschitz derivative:*

$$|\phi_j'(s) - \phi_j'(s')| \leq L_{\phi_j}|s - s'|, \qquad s, s' \in \mathbf{R}. \tag{31}$$

Then f_j has a block-Lipschitz gradient with constants

$$L_{ji} = L_{\phi_j}\left(\|A_{ji}^T\|^*_{(i)}\right)^2, \qquad i = 1, 2, \ldots, n. \tag{32}$$

In other words, f_j satisfies (16) *with constants L_{ji} given above.*

Proof. For any $x \in \mathbf{R}^N$, $t \in \mathbf{R}^{N_i}$, and i we have

$$\begin{aligned}
\|\nabla_i f_j(x + U_i t) - \nabla_i f_j(x)\|^*_{(i)} &\overset{(4)}{=} \|U_i^T(e_j^T A)^T\left(\phi_j'(e_j^T A(x + U_i t)) - \phi_j'(e_j^T Ax)\right)\|^*_{(i)} \\
&\overset{(30)}{=} \|A_{ji}^T\phi_j'(e_j^T A(x + U_i t)) - A_{ji}^T\phi_j'(e_j^T Ax)\|^*_{(i)} \\
&\leq \|A_{ji}^T\|^*_{(i)}|\phi_j'(e_j^T A(x + U_i t)) - \phi_j'(e_j^T Ax)| \\
&\overset{(31)+(30)}{\leq} \|A_{ji}^T\|^*_{(i)}L_{\phi_j}|A_{ji}t| \;\leq\; \|A_{ji}^T\|^*_{(i)}L_{\phi_j}\|A_{ji}^T\|^*_{(i)}\|t\|_{(i)},
\end{aligned}$$

where the last step follows by applying the Cauchy–Schwarz inequality. ∎

EXAMPLE 2 (quadratics). *Consider the quadratic function*

$$f(x) = \frac{1}{2}\|Ax - b\|^2 = \frac{1}{2}\sum_{j=1}^m (e_j^T Ax - b_j)^2.$$

Then $f_j(x) = \phi_j(e_j^T Ax)$, where $\phi_j(s) = \frac{1}{2}(s - b_j)^2$ and $L_{\phi_j} = 1$.

Table 4 *Lipschitz constants of the derivative of selected scalar loss functions.*

Loss	$\phi(s)$	L_ϕ
Square loss	$s^2/2$	1
Logistic loss	$\log(1+e^s)$	1/4

Table 5 *ESO stepsizes for coordinate descent methods suggested in the literature in the case of a quadratic $f(x) = \frac{1}{2}\|Ax-b\|^2$. For simplicity, we consider the setup with elementary block sizes ($N_i = 1$) and the absolute value norm (this corresponds to $B_i = 1$).*

Paper	v_i
Richtárik and Takáč [33]	$v_i^{\rm rt} = \sum_{j=1}^m \left(1 + \frac{(\omega-1)(\tau-1)}{\max\{1,n-1\}}\right) A_{ji}^2$
Necoara and Clipici [19]	$v_i^{\rm nc} = \sum_{j:i\in C_j} \sum_{k=1}^n A_{jk}^2$
This paper	$v_i^{\rm fr} = \sum_{j=1}^m \left(1 + \frac{(\omega_j-1)(\tau-1)}{\max\{1,n-1\}}\right) A_{ji}^2$

(i) *Choose $n = 1$; that is, all coordinates belong to a single block only. Further, let B_1 (recall that B_i is the positive definite matrix defining the norm associated with block i) be the $N \times N$ identity matrix. Then $L_{j1} = L_{\phi_j}(\|A_{j:}^T\|_{(1)}^*)^2 = \sum_{i=1}^n A_{ji}^2$.*

(ii) *Consider the block setup with $N_i = 1$ (all blocks are of size 1) and $B_i = 1$ for all $i \in [n]$. Then $L_{ji} = L_{\phi_j}(\|A_{ji}^T\|_{(i)}^*)^2 = A_{ji}^2$.*

(iii) *Choose nontrivial block sizes and define data-driven block norms with $B_i = A_i^T A_i$, where $A_i = AU_i$, assuming that the matrices $A_i^T A_i$ are positive definite (necessarily, $N_i \le m$). The idea here is that data-driven norms better capture the curvature of the function in the subspaces spanned by the blocks. Then*

$$L_{ji} = L_{\phi_j}(\|A_{ji}^T\|_{(i)}^*)^2 \overset{(6)}{=} \langle (A_i^T A_i)^{-1} A_{ji}^T, A_{ji}^T\rangle \overset{(30)}{=} e_j^T M_i e_j,$$

where $M_i \overset{\rm def}{=} A_i(A_i^T A_i)^{-1}A_i^T \in \mathbf{R}^{m\times m}$. Since M_i is a projection matrix, all its eigenvalues are either 0 or 1, and $\mathrm{tr}(M_i) = \mathrm{rank}(A_i) = N_i$. In particular, its diagonal elements, L_{ji}, satisfy $0 \le L_{ji} \le 1$ and $\sum_j L_{ji} = N_i$.

Table 4 lists constants L_ϕ for selected scalar loss functions ϕ popular in machine learning. In Table 5 we list stepsizes for coordinate descent methods proposed in the literature. For simplicity of comparison, this is done for the setup described in case (i) in the above example. It can be seen that our stepsizes are better than those proposed by Richtárik and Takáč [33] and those proposed by Necoara and Clipici [19]. Indeed, $v_i^{\rm rt} \ge v_i^{\rm fr}$ for all i. The difference grows as τ grows, and there is equality for $\tau = 1$. We also have $\|v^{\rm nc}\|_1 \ge \|v^{\rm fr}\|_1$, but here the difference decreases with τ, and there is equality for $\tau = n$.

4. Accelerated Parallel and Proximal Coordinate Descent. We are interested in solving the regularized optimization problem

$$\begin{aligned} &\text{minimize} && F(x) \overset{\rm def}{=} f(x) + \psi(x) \\ &\text{subject to} && x = (x^{(1)},\dots,x^{(n)}) \in \mathbf{R}^{N_1}\times\cdots\times\mathbf{R}^{N_n} = \mathbf{R}^N, \end{aligned} \tag{33}$$

where $\psi : \mathbf{R}^N \to \mathbf{R} \cup \{+\infty\}$ is a (possibly nonsmooth) convex regularizer that is separable in the blocks $x^{(i)}$:

$$\psi(x) = \sum_{i=1}^{n} \psi_i(x^{(i)}). \tag{34}$$

The functions $\psi_i : \mathbf{R}^{N_i} \to \mathbf{R} \cup \{+\infty\}$ are assumed to be convex and closed (i.e., lower semicontinuous).

4.1. The Algorithm. We now describe our method (Algorithm 1). It is presented here in a form that facilitates analysis and comparison with existing methods. In section 6 we rewrite the method in a different (equivalent) form—one that is geared toward practical efficiency.

Algorithm 1 APPROX: **A**ccelerated **P**arallel and **Prox**imal Coordinate Descent

1: Choose $x_0 \in \mathbf{R}^N$ and set $z_0 = x_0$ and $\theta_0 = \frac{\tau}{n}$
2: **for** $k \geq 0$ **do**
3: $y_k = (1 - \theta_k)x_k + \theta_k z_k$
4: Generate a random set of blocks $S_k \sim \hat{S}$
5: $z_{k+1} = z_k$
6: **for** $i \in S_k$ **do**
7: $z_{k+1}^{(i)} = \arg\min_{z \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(y_k), z - y_k^{(i)} \rangle + \frac{n\theta_k v_i}{2\tau} \|z - z_k^{(i)}\|_{(i)}^2 + \psi_i(z) \right\}$
8: **end for**
9: $x_{k+1} = y_k + \frac{n}{\tau}\theta_k(z_{k+1} - z_k)$
10: $\theta_{k+1} = \frac{\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2}{2}$
11: **end for**

The method starts with $x_0 \in \mathbf{R}^N$ and generates three vector sequences denoted $\{x_k, y_k, z_k\}_{k \geq 0}$. In step 3, y_k is defined as a convex combination of x_k and z_k, which may in general be full-dimensional vectors. This is not efficient, but we will ignore this issue for now. In section 6 we show that it is possible to implement the method in such a way that it not necessary to ever form y_k. In step 4 we generate a random block sampling S_k and then perform steps 5–9 in parallel. The assignment $z_{k+1} \leftarrow z_k$ is not necessary in practice; the vector z_k should be overwritten in place. Instead, steps 5–8 should be seen as saying that we update blocks $i \in S_k$ of z_k by solving $|S_k|$ proximal problems in parallel, and we call the resulting vector z_{k+1}. Note in step 9, x_{k+1} should also be computed in parallel. Indeed, x_{k+1} is obtained from y_k by changing the blocks of y_k that belong to S_k; this is because z_{k+1} and z_k differ in those blocks only. Note that gradients are evaluated only at y_k. We show in section 6 how this can be done efficiently for some problems, without the need to form y_k.

We now formulate the main result of this paper; its proof is in section 5.

THEOREM 3. *Let Assumption* 1 *be satisfied, with* $(f, \hat{S}) \sim \mathrm{ESO}(v)$, *where* $\tau = \mathbf{E}[|\hat{S}|] > 0$. *Let* $x_0 \in \operatorname{dom} \psi$, *and assume that the random sets* S_k *in Algorithm* 1 *are chosen independently, following the distribution of* $\hat{S}$. *Let* x_* *be any optimal point of problem* (33). *Then the iterates* $\{x_k\}_{k \geq 1}$ *of APPROX satisfy*

$$\mathbf{E}[F(x_k) - F(x_*)] \leq \frac{4n^2 C_*}{((k-1)\tau + 2n)^2}, \tag{35}$$

where

$$C_* \stackrel{\text{def}}{=} \left(1 - \frac{\tau}{n}\right)(F(x_0) - F(x_*)) + \frac{1}{2}\|x_0 - x_*\|_v^2. \tag{36}$$

In other words, for any $0 < \epsilon \leq C_*$, the number of iterations for obtaining an ϵ-solution in expectation does not exceed

$$k = \left\lceil \frac{2n}{\tau}\left(\sqrt{\frac{C_*}{\epsilon}} - 1\right) + 1 \right\rceil. \tag{37}$$

Let us now comment on the result.

Assumptions. For the complexity result to hold, we do not assume that f is of the form (1)—all that is needed is Assumption 1.

All Coordinates Belong to a Single Block. If we choose $n = 1$ (single block), then the only reasonable sampling is to pick this block with probability 1 ($\mathbf{P}(\hat{S} = \{1\}) = 1$). The method becomes deterministic. Let B_1 be the $N \times N$ identity matrix, so that $\|\cdot\|_{(1)}$ is the standard Euclidean norm (and hence $\|x\|_v^2 = v\|x\|^2$). In this case we recover Tseng's accelerated proximal gradient descent [47], and the complexity bound (35) takes the form

$$F(x_k) - F(x_*) \leq \frac{2v\|x_0 - x_*\|^2}{(k+1)^2}, \tag{38}$$

where v is the Lipschitz constant of the gradient of f (this is what the assumption $(f, \hat{S}) \sim ESO(v)$ means for this sampling). Note that Theorem 1 gives a bound on the Lipschitz constant for f of the form (2) satisfying Assumption 2.

Updating All Blocks. In the case when we update all blocks in one iteration ($\tau = n$), the method is deterministic, and the bound (35) simplifies to

$$F(x_k) - F(x_*) \leq \frac{2\|x_0 - x_*\|_v^2}{(k+1)^2} = \frac{2\frac{\|v\|_1}{n}\|x_0 - x_*\|_{\tilde{v}}^2}{(k+1)^2}, \tag{39}$$

where, as before, $\tilde{v} = nv/\|v\|_1$. Note that $\|\cdot\|_{\tilde{v}}$ is a weighted norm with weights adding up to n; which means it is "comparable" to the standard Euclidean norm (all weights of which are equal to 1 and hence sum up to n). If we use stepsize v proposed in Theorem 1, then in view of part (ii) of that theorem, bound (39) takes the form

$$F(x_k) - F(x_*) \leq \frac{2\bar{\omega}\bar{L}\|x_0 - x_*\|_w^2}{(k+1)^2}, \tag{40}$$

as advertised in the abstract. Recall that $\bar{\omega}$ is a data-weighted *average* of the values $\{\omega_j\}$ and that $\sum_i w_i = n$. In contrast, using the stepsizes proposed by Richtárik and Takáč [33] (see Table 5), we get

$$F(x_k) - F(x_*) \leq \frac{2\omega\frac{\sum_i L_i}{n}\|x_0 - x_*\|_{\tilde{v}}^2}{(k+1)^2}. \tag{41}$$

Note that in the case when the functions f_j are convex quadratics ($f_j(x) = \frac{1}{2}(a_j^T x - b_j)^2$), for instance, we have $L_i = \sum_j L_{ji}$, and hence the new stepsizes lead to a vast improvement in the complexity in cases when $\bar{\omega} \ll \omega$. On the other hand, in cases where $L_i \ll \sum_j L_{ji}$ (which can happen with logistic regression, for instance), the result based on the Richtárik–Takáč stepsizes [33] may be better.

LASSO. We now illustrate our complexity results on the LASSO (L1 regularized least-squares) problem, which is of the form (1) with

$$f(x) = \frac{1}{2}\|Ax - b\|^2, \quad \psi(x) = \lambda\|x\|_1, \quad f_j(x) = \frac{1}{2}(A_{j:}x - b_j)^2,$$

where $A \in \mathbf{R}^{m\times N}$, $b \in \mathbf{R}^m$, and $\lambda > 0$. If $N \gg m$, the state-of-the-art method for solving LASSO is (nonaccelerated) coordinate descent [38, 32]. As we have seen, the existing accelerated coordinate descent method of Nesterov [23] requires full-dimensional operations in each iteration, which makes the method much less efficient than standard coordinate descent. However, APPROX does not suffer from this issue; we will show in section 6 that the average cost of a single iteration of APPROX (for $n = N$) is proportional to $\mathrm{nnz}(A)\tau/N$.

In APPROX we have certain design parameters to decide on. First, we can choose the number of blocks (n), then we decide how to partition the N coordinates into these blocks, and, finally, we decide how many (τ) of these blocks we update in a single iteration. Let $K(n,\tau)$ be the total complexity of APPROX specialized to n blocks and τ block updates per iteration for obtaining an ϵ-solution. For simplicity, assume $x_0 = 0$. By setting $n = 1$ (and hence $\tau = 1$), APPROX specializes to accelerated (proximal) gradient descent (see Table 6), the cost of a single iteration is proportional to the number of nonzeros in A ($\mathrm{nnz}(A)$), and we have

$$K(1,1) \overset{(38)}{=} \frac{4\,\mathrm{nnz}(A)\sqrt{v}\|x_*\|}{\sqrt{\epsilon}} = \frac{4\,\mathrm{nnz}(A)\sqrt{\sum_{i=1}^N v(x_*^i)^2}}{\sqrt{\epsilon}},$$

where v is the Lipschitz constant of the gradient of f and hence can be chosen to be $\lambda_{\max}(A^TA)$ or $\sum_{j=1}^m \omega_j \sum_{i=1}^N A_{ji}^2$ (an efficiently computable upper bound on $\lambda_{\max}(A^TA)$ which we obtain in Theorem 1(ii)). The former bound is better than the latter, but requires more preprocessing work for the computation of v (which affects the stepsizes of the method).

Let us now compare these bounds with what we obtain for APPROX with the setting $n = N$ and $\tau = N$ (see Table 6):

$$K(N,N) = \frac{4\,\mathrm{nnz}(A)\|x_*\|_v}{\sqrt{\epsilon}} = \frac{4\,\mathrm{nnz}(A)\sqrt{\sum_{i=1}^N v_i(x_*^i)^2}}{\sqrt{\epsilon}}, \tag{42}$$

where $v_i = \sum_{j=1}^m \omega_j A_{ji}^2$. Notice that $K(N,N)$ can be much better than $K(1,1)$. Indeed, if the data is sufficiently sparse (parameters ω_j being sufficiently small), and if the largest eigenvalue of A^TA is close to its trace, then

$$\sum_{j=1}^m \omega_j A_{ji}^2 \ll \sum_{i=1}^N \sum_{j=1}^m A_{ji}^2 = \mathrm{tr}(A^TA) \approx \lambda_{\max}(A^TA),$$

whence $K(N,N) \ll K(1,1)$.

Finally, let us compare APPROX with $n = N$ and $\tau = 1$ with standard coordinate descent (again, see Table 6). We can observe that APPROX will be better as soon as $k \geq 8N$. Indeed, after k iterations of APPROX, coordinate descent has done twice as many iterations and the worst-case complexity bounds B_{approx} and B_{cd} compare as

$$B_{\mathrm{approx}} \approx \frac{4N^2C_*}{k^2} = \frac{8N}{k} \times \frac{NC_*}{2k} \approx \frac{8N}{k} B_{\mathrm{cd}}.$$

Table 6 *Complexity of coordinate descent, accelerated stochastic dual coordinate ascent (SDCA) and selected variants of APPROX, when applied to the LASSO problem:* $\min_{x\in\mathbf{R}^N} \frac{1}{2}\|Ax-b\|^2+\lambda\|x\|_1$, *where* $A\in\mathbf{R}^{m\times N}$. *APPROX is superior to both coordinate descent and accelerated SDCA. For simplicity, we assume the starting point is* $x_0=0$ *and define* $C_*=2(1-\frac{\tau}{n})(F(0)-F(x_*))+\|x_*\|_v^2$ *as in Theorem* 3. *The complexity bounds for APPROX are exact; we omit constant terms in the complexity bounds for standard coordinate descent, accelerated SDCA, and in the formulas for cost of* 1 *iteration. The notation* $\|\cdot\|_v$ *means a weighted Euclidean norm with weights defined by the vector* $v=(v_1,\dots,v_n)$. *Each algorithm depends on a norm defined in the last column. Note that the norms can differ a lot.*

Method	Cost of 1 iteration	Complexity	v_i
Coord. descent [32]	$\frac{\mathrm{nnz}(A)}{N}$	$\frac{NC_*}{\epsilon}$	$\sum_{j=1}^m A_{ji}^2$
APPROX ($n=N,\tau=1$)	$\frac{2\,\mathrm{nnz}(A)}{N}$	$\frac{2N\sqrt{C_*}}{\sqrt{\epsilon}}$	$\sum_{j=1}^m A_{ji}^2$
Accelerated SDCA [41] ($n=N,\tau=1$)	$\mathrm{nnz}(A)\log\left(\frac{v\|x_*\|^2}{\epsilon}\right)$	$\frac{4\sqrt{2v}\|x_*\|}{\sqrt{\epsilon}}\log\left(\frac{v\|x_*\|^2}{\epsilon}\right)$	$\max_i \sum_{j=1}^m A_{ji}^2$
APPROX ($n=1,\tau=1$) = Acc. Gradient Descent	$2\,\mathrm{nnz}(A)$	$\frac{2\sqrt{v}\|x_*\|}{\sqrt{\epsilon}}$	$\lambda_{\max}(A^TA)$ or $\sum_{i=1}^N\sum_{j=1}^m \omega_j A_{ji}^2$
APPROX ($n=N,\tau=N$)	$2\,\mathrm{nnz}(A)$	$\frac{2\|x_*\|_v}{\sqrt{\epsilon}}$	$\sum_{j=1}^m \omega_j A_{ji}^2$

Less Aggressive Choice of θ_k. Instead of θ_k, one may consider any sequence such that $(1-\theta'_{k+1})/(\theta'_{k+1})^2\le 1/(\theta'_k)^2$ and $\theta'_0\le\tau/n$; see [47]. Note that in this case, one should replace θ_k^2 by $(\theta'_0)^2\prod_{l=1}^k(1-\theta'_l)$ in Algorithm 2.

5. Complexity Analysis.

In this section we prove Theorem 3.

5.1. Four Lemmas.

In the first lemma we summarize well-known properties of the sequence θ_k used in Algorithm 1.

LEMMA 1 (Tseng [47]). *The sequence* $\{\theta_k\}_{k\ge0}$ *defined in Algorithm* 1 *is decreasing and satisfies* $0<\theta_k\le\frac{2}{k+2n/\tau}\le\frac{\tau}{n}\le 1$ *and*

$$\frac{1-\theta_{k+1}}{\theta_{k+1}^2}=\frac{1}{\theta_k^2}. \tag{43}$$

We now give an explicit characterization of x_k as a convex combination of the vectors $z_0,\dots,z_k$.

LEMMA 2. *Let* $\{x_k,z_k\}_{k\ge0}$ *be the iterates of Algorithm* 1. *Then, for all* $k\ge0$,

$$x_k=\sum_{l=0}^k \gamma_k^l z_l, \tag{44}$$

where the coefficients $\gamma_k^0,\gamma_k^1,\dots,\gamma_k^k$ *are nonnegative and sum to* 1. *That is,* x_k *is a convex combination of the vectors* $z_0,z_1,\dots,z_k$. *In particular, the constants are*

defined recursively in k by setting $\gamma_0^0 = 1$, $\gamma_1^0 = 0$, $\gamma_1^1 = 1$ and for $k \geq 1$,

$$\gamma_{k+1}^l = \begin{cases} (1-\theta_k)\gamma_k^l, & l = 0, \dots, k-1, \\ \theta_k(1 - \frac{n}{\tau}\theta_{k-1}) + \frac{n}{\tau}(\theta_{k-1} - \theta_k), & l = k, \\ \frac{n}{\tau}\theta_k, & l = k+1. \end{cases} \tag{45}$$

Moreover, for all $k \geq 0$, the following identity holds:

$$\gamma_{k+1}^k + \frac{n-\tau}{\tau}\theta_k = (1-\theta_k)\gamma_k^k. \tag{46}$$

Proof. We proceed by induction. First, notice that $x_0 = z_0 = \gamma_0^0 z_0$. This implies that $y_0 = z_0$, which in turn, together with $\theta_0 = \frac{\tau}{n}$, gives $x_1 = y_0 + \frac{n}{\tau}\theta_0(z_1 - z_0) = z_1 = \gamma_1^0 z_0 + \gamma_1^1 z_1$. Assuming now that (44) holds for some $k \geq 1$, we obtain

$$\begin{aligned} x_{k+1} &\overset{\text{(Alg 1, step 9)}}{=} y_k + \frac{n}{\tau}\theta_k(z_{k+1} - z_k) \\ &\overset{\text{(Alg 1, step 3)}}{=} (1-\theta_k)x_k + \theta_k z_k - \frac{n}{\tau}\theta_k z_k + \frac{n}{\tau}\theta_k z_{k+1} \\ &= \sum_{l=0}^{k-1} \underbrace{(1-\theta_k)\gamma_k^l}_{\gamma_{k+1}^l} z_l + \underbrace{\left((1-\theta_k)\gamma_k^k + \theta_k - \frac{n}{\tau}\theta_k\right)}_{\gamma_{k+1}^k} z_k + \underbrace{\left(\frac{n}{\tau}\theta_k\right)}_{\gamma_{k+1}^{k+1}} z_{k+1}. \end{aligned} \tag{47}$$

By applying Lemma 1, together with the inductive assumption that $\gamma_k^l \geq 0$ for all l, we observe that $\gamma_{k+1}^l \geq 0$ for all l. It remains to show that the constants sum to 1. This is true since x_k is a convex combination of $z_1, \dots, z_k$, and by (47), x_{k+1} is an affine combination of x_k, z_k, and z_{k+1}. ∎

Define

$$\begin{aligned} \tilde{z}_{k+1} &\overset{\text{def}}{=} \arg\min_{z \in \mathbf{R}^N} \left\{ \psi(z) + \langle \nabla f(y_k), z - y_k \rangle + \frac{n\theta_k}{2\tau}\|z - z_k\|_v^2 \right\} \\ &\overset{(5)+(34)}{=} \arg\min_{z=(z^{(1)},\dots,z^{(n)}) \in \mathbf{R}^N} \sum_{i=1}^n \left\{ \psi_i(z^{(i)}) + \langle \nabla_i f(y_k), z^{(i)} - y_k^{(i)} \rangle + \frac{n\theta_k v_i}{2\tau}\|z^{(i)} - z_k^{(i)}\|_{(i)}^2 \right\}. \end{aligned}$$

From this and the definition of z_{k+1} we see that

$$z_{k+1}^{(i)} = \begin{cases} \tilde{z}_{k+1}^{(i)}, & i \in S_k, \\ z_k^{(i)}, & i \notin S_k. \end{cases} \tag{48}$$

The next lemma is an application to a specific function of a well-known result that can be found, for instance, in [47, Property 1]. The result was used by Tseng to construct a simplified complexity proof for a proximal gradient descent method.

LEMMA 3 (see [47]). *Let $\xi(u) \overset{\text{def}}{=} f(y_k) + \langle \nabla f(y_k), u - y_k \rangle + \frac{n\theta_k}{2\tau}\|u - z_k\|_v^2$. Then*

$$\psi(\tilde{z}_{k+1}) + \xi(\tilde{z}_{k+1}) \leq \psi(x_*) + \xi(x_*) - \frac{n\theta_k}{2\tau}\|x_* - \tilde{z}_{k+1}\|_v^2. \tag{49}$$

Our next lemma is a technical result connecting the gradient mapping (producing $\tilde{z}_{k+1}$) and the randomized block gradient mapping (producing the random vector z_{k+1}). The lemma reduces to a trivial identity in the case when of a single block ($n = 1$). From now on, by $\mathbf{E}_k$ we denote the expectation with respect to S_k, keeping everything else fixed.

LEMMA 4. *For any $x \in \mathbf{R}^N$ and $k \geq 0$,*

$$\mathbf{E}_k\left[\|z_{k+1}-x\|_v^2 - \|z_k - x\|_v^2\right] = \frac{\tau}{n}\left(\|\tilde{z}_{k+1}-x\|_v^2 - \|z_k-x\|_v^2\right). \tag{50}$$

Moreover,

$$\mathbf{E}_k\left[\psi(z_{k+1})\right] = \left(1-\frac{\tau}{n}\right)\psi(z_k) + \frac{\tau}{n}\psi(\tilde{z}_{k+1}). \tag{51}$$

Proof. Let $\hat{S}$ be any uniform sampling and $a, h \in \mathbf{R}^N$. By Theorem 4 in [33],

$$\begin{aligned} \mathbf{E}[\|h_{[\hat{S}]}\|_v^2] = \frac{\tau}{n}\|h\|_v^2, \qquad \mathbf{E}[\langle a, h_{[\hat{S}]}\rangle_v] = \frac{\tau}{n}\langle a, h\rangle_v, \\ \mathbf{E}[\psi(a+h_{[\hat{S}]})] = \left(1-\frac{\tau}{n}\right)\psi(a) + \frac{\tau}{n}\psi(a+h), \end{aligned} \tag{52}$$

where $\langle a, h\rangle_v \overset{\text{def}}{=} \sum_{i=1}^n v_i \langle a^{(i)}, h^{(i)}\rangle$. Let $h = \tilde{z}_{k+1} - z_k$. In view of (3) and (48), we can write $z_{k+1} - z_k = h_{[S_k]}$. Applying the first two identities in (52) with $a = z_k - x$ and $\hat{S} = S_k$, we get

$$\begin{aligned} \mathbf{E}_k\left[\|z_{k+1}-x\|_v^2 - \|z_k-x\|_v^2\right] &= \mathbf{E}_k\left[\|h_{[S_k]}\|_v^2 + 2\langle z_k - x, h_{[S_k]}\rangle_v\right] \\ &\overset{(52)}{=} \frac{\tau}{n}\left(\|h\|_v^2 + 2\langle z_k - x, h\rangle_v\right) = \frac{\tau}{n}\left(\|\tilde{z}_{k+1}-x\|_v^2 - \|z_k-x\|_v^2\right). \end{aligned} \tag{53}$$

The remaining statement follows from the last identity in (52) used with $a = z_k$. ∎

5.2. Proof of Theorem 3. Using Lemma 2 and convexity of ψ,

$$\psi(x_k) \overset{(44)}{=} \psi\left(\sum_{l=0}^k \gamma_k^l z_l\right) \overset{\text{(convexity)}}{\leq} \sum_{l=0}^k \gamma_k^l \psi(z_l) \overset{\text{def}}{=} \hat{\psi}_k, \tag{54}$$

which holds for all $k \geq 0$. From this we get

$$\begin{aligned} \mathbf{E}_k[\hat{\psi}_{k+1}] &\overset{(54)+(45)}{=} \sum_{l=0}^k \gamma_{k+1}^l \psi(z_l) + \frac{n}{\tau}\theta_k \mathbf{E}_k\left[\psi(z_{k+1})\right] \\ &\overset{(51)}{=} \sum_{l=0}^k \gamma_{k+1}^l \psi(z_l) + \frac{n}{\tau}\theta_k\left(\left(1-\frac{\tau}{n}\right)\psi(z_k) + \frac{\tau}{n}\psi(\tilde{z}_{k+1})\right) \\ &= \sum_{l=0}^k \gamma_{k+1}^l \psi(z_l) + \left(\frac{n}{\tau}-1\right)\theta_k\psi(z_k) + \theta_k\psi(\tilde{z}_{k+1}). \end{aligned} \tag{55}$$

Since $x_{k+1} = y_k + h_{[S_k]}$ with $h = \frac{n}{\tau}\theta_k(\tilde{z}_{k+1} - z_k)$, we can use ESO to bound

$$\begin{aligned} \mathbf{E}_k[f(x_{k+1})] &\overset{(15)}{\leq} f(y_k) + \theta_k\langle \nabla f(y_k), \tilde{z}_{k+1} - z_k\rangle + \frac{n\theta_k^2}{2\tau}\|\tilde{z}_{k+1}-z_k\|_v^2 \\ &= (1-\theta_k)f(y_k) - \theta_k\langle \nabla f(y_k), z_k - y_k\rangle \\ &\quad + \theta_k\Big(f(y_k) + \langle \nabla f(y_k), \tilde{z}_{k+1} - y_k\rangle + \frac{n\theta_k}{2\tau}\|\tilde{z}_{k+1}-z_k\|_v^2\Big). \end{aligned} \tag{56}$$

Note that from the definition of y_k in the algorithm, we have

$$\theta_k(y_k - z_k) = ((1-\theta_k)x_k - y_k) + \theta_k y_k = (1-\theta_k)(x_k - y_k). \tag{57}$$

For all $k \geq 0$ we define an upper bound on $F(x_k)$,

$$\hat{F}_k \overset{\text{def}}{=} \hat{\psi}_k + f(x_k) \overset{(54)}{\geq} F(x_k), \tag{58}$$

and bound the expectation of $\hat{F}_{k+1}$ in S_k as follows:

$$\begin{aligned}
\mathbf{E}_k[\hat{F}_{k+1}] &= \mathbf{E}_k[\hat{\psi}_{k+1}] + \mathbf{E}_k[f(x_{k+1})] \\
&\overset{(55)+(56)}{\leq} \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \frac{n-\tau}{\tau}\theta_k\psi(z_k) + (1-\theta_k)f(y_k) - \theta_k\langle \nabla f(y_k), z_k - y_k\rangle \\
&\quad + \theta_k\Big(\psi(\tilde{z}_{k+1}) + f(y_k) + \langle \nabla f(y_k), \tilde{z}_{k+1} - y_k\rangle + \frac{n\theta_k}{2\tau}\|\tilde{z}_{k+1} - z_k\|_v^2\Big) \\
&\overset{(49)}{\leq} \sum_{l=0}^{k} \gamma_{k+1}^l \psi(z_l) + \frac{n-\tau}{\tau}\theta_k\psi(z_k) + (1-\theta_k)f(y_k) - \theta_k\langle \nabla f(y_k), z_k - y_k\rangle \\
&\quad + \theta_k\Big(\psi(x_*) + f(y_k) + \langle \nabla f(y_k), x_* - y_k\rangle \\
&\qquad + \frac{n\theta_k}{2\tau}(\|x_* - z_k\|_v^2 - \|x_* - \tilde{z}_{k+1}\|_v^2)\Big).
\end{aligned} \tag{59}$$

Using (57), we can now further bound (59) as follows:

$$\begin{aligned}
\mathbf{E}_k[\hat{F}_{k+1}] &\overset{(59)+(57)}{\leq} \sum_{l=0}^{k-1} \underbrace{\gamma_{k+1}^l}_{\overset{(45)}{=}(1-\theta_k)\gamma_k^l} \psi(z_l) + \underbrace{\left(\gamma_{k+1}^k + \frac{n-\tau}{\tau}\theta_k\right)}_{\overset{(46)}{=}(1-\theta_k)\gamma_k^k}\psi(z_k) \\
&\quad + \underbrace{(1-\theta_k)f(y_k) + (1-\theta_k)\langle \nabla f(y_k), x_k - y_k\rangle}_{\leq (1-\theta_k)f(x_k)} \\
&\quad + \theta_k\Big(\underbrace{\psi(x_*) + f(y_k) + \langle \nabla f(y_k), x_* - y_k\rangle}_{\leq F(x_*)}\Big) \\
&\qquad + \frac{n\theta_k^2}{2\tau}\|x_* - z_k\|_v^2 - \frac{n\theta_k^2}{2\tau}\|x_* - \tilde{z}_{k+1}\|_v^2 \\
&\overset{(54)+(58)}{\leq} (1-\theta_k)\hat{F}_k + \theta_k F(x_*) + \frac{n\theta_k^2}{2\tau}\left(\|x_* - z_k\|_v^2 - \|x_* - \tilde{z}_{k+1}\|_v^2\right) \\
&\overset{(50)}{=} (1-\theta_k)\hat{F}_k + \theta_k F(x_*) + \frac{n^2\theta_k^2}{2\tau^2}\left(\|x_* - z_k\|_v^2 - \mathbf{E}_k\left[\|x_* - z_{k+1}\|_v^2\right]\right).
\end{aligned}$$

Dividing both sides in the last inequality by θ_k^2, using (43), and rearranging the terms, we obtain

$$\frac{1-\theta_{k+1}}{\theta_{k+1}^2}\mathbf{E}_k[\hat{F}_{k+1} - F(x_*)] + \frac{n^2}{2\tau^2}\mathbf{E}_k[\|x_* - z_{k+1}\|_v^2] \leq \frac{1-\theta_k}{\theta_k^2}(\hat{F}_k - F(x_*)) + \frac{n^2}{2\tau^2}\|x_* - z_k\|_v^2.$$

We now apply expectation to the above inequality and unroll the recurrence, obtaining

$$\frac{1-\theta_k}{\theta_k^2}\mathbf{E}[\hat{F}_k - F(x_*)] + \frac{n^2}{2\tau^2}\mathbf{E}[\|x_* - z_k\|_v^2] \leq \frac{1-\theta_0}{\theta_0^2}(\hat{F}_0 - F(x_*)) + \frac{n^2}{2\tau^2}\|x_* - z_0\|_v^2, \tag{60}$$

from which we finally get, for $k \geq 1$,

$$\begin{aligned}\mathbf{E}[F(x_k) - F(x_*)] &\overset{(58)}{\leq} \mathbf{E}[\hat{F}_k - F(x_*)] \\ &\overset{(60)}{\leq} \frac{\theta_{k-1}^2}{\theta_0^2}(1-\theta_0)(\hat{F}_0 - F(x_*)) + \frac{n^2\theta_{k-1}^2}{2\tau^2}\|x_* - z_0\|_v^2 \\ &\leq \frac{4n^2}{((k-1)\tau + 2n)^2}\left(\left(1 - \frac{\tau}{n}\right)(F(x_0) - F(x_*)) + \frac{1}{2}\|x_0 - x_*\|_v^2\right),\end{aligned}$$

where in the last step we have used the facts that $\hat{F}_0 = F(x_0)$, $x_0 = z_0$, $\theta_0 = \frac{\tau}{n}$ and the estimate $\theta_{k-1} \leq \frac{2}{k-1+2n/\tau}$ from Lemma 1.

6. Implementation without Full-Dimensional Vector Operations. Algorithm 1, as presented, performs full-dimensional vector operations. Indeed, y_k is defined as a convex combination of x_k and z_k. Also, x_{k+1} is obtained from y_k by changing $|S_k|$ coordinates; however, if $|S_k|$ is small, the latter operation is not costly. In any case, vectors x_k and z_k will in general be dense, and hence computation of y_k may cost $O(N)$ arithmetic operations. However, simple (i.e., nonaccelerated) coordinate descent methods are successful and popular precisely because they can avoid such operations. Adapting ideas from Lee and Sidford [13], we rewrite[9] Algorithm 1 into a new form, incarnated as Algorithm 2.

Algorithm 2 APPROX (written in a form facilitating efficient implementation)

1: Pick $\tilde{z}_0 \in \mathbf{R}^N$ and set $\theta_0 = \frac{\tau}{n}$, $u_0 = 0$
2: **for** $k \geq 0$ **do**
3: Generate a random set of blocks $S_k \sim \hat{S}$
4: $u_{k+1} \leftarrow u_k$, $\tilde{z}_{k+1} \leftarrow \tilde{z}_k$
5: **for** $i \in S_k$ **do**
6: $t_k^{(i)} = \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(\theta_k^2 u_k + \tilde{z}_k), t\rangle + \frac{n\theta_k v_i}{2\tau}\|t\|_{(i)}^2 + \psi_i(\tilde{z}_k^{(i)} + t) \right\}$
7: $\tilde{z}_{k+1}^{(i)} \leftarrow \tilde{z}_k^{(i)} + t_k^{(i)}$
8: $u_{k+1}^{(i)} \leftarrow u_k^{(i)} - \frac{1 - \frac{n}{\tau}\theta_k}{\theta_k^2} t_k^{(i)}$
9: **end for**
10: $\theta_{k+1} = \frac{\sqrt{\theta_k^4 + 4\theta_k^2} - \theta_k^2}{2}$
11: **end for**
12: OUTPUT: $\theta_k^2 u_{k+1} + \tilde{z}_{k+1}$

Note that if instead of updating the constants θ_k as in line 10 we keep them constant throughout, $\theta_k = \frac{\tau}{n}$, then $u_k = 0$ for all k. The resulting method is precisely the PCDM algorithm (*nonaccelerated* parallel block coordinate descent method) proposed and analyzed in [33].

As it is not immediately obvious that the two methods (Algorithms 1 and 2) are equivalent, we include the following result. Its proof can be found in the appendix.

PROPOSITION 1 (equivalence). *Algorithms* 1 *and* 2 *are equivalent. In particular, if we run Algorithm* 2 *with* $\tilde{z}_0 = x_0$, *where* $x_0 \in \operatorname{dom}\psi$ *is the starting point of*

[9] Note that we override the notation $\tilde{z}_k$ here—it now has a different meaning from that in section 5.

Algorithm 1*, and define*

$$\tilde{x}_k \overset{\text{def}}{=} \begin{cases} \tilde{z}_0, & k = 0, \\ \theta_{k-1}^2 u_k + \tilde{z}_k, & k \geq 1, \end{cases} \tag{61}$$

$$\tilde{y}_k \overset{\text{def}}{=} \theta_k^2 u_k + \tilde{z}_k, \quad k \geq 0, \tag{62}$$

then $x_k = \tilde{x}_k$*,* $y_k = \tilde{y}_k$*, and* $z_k = \tilde{z}_k$ *for all* $k \geq 0$*.*

In Algorithm 2 we never need to form x_k throughout the iterations. The only time this is needed is when producing the output: $x_{k+1} = \theta_k^2 u_{k+1} + z_{k+1}$. More importantly, the method does need to explicitly compute y_k. Instead, we introduce a new vector, u_k, and express y_k as $y_k = \theta_k^2 u_k + \tilde{z}_k$. The method accesses y_k only via the block gradients $\nabla_i f(y_k)$ for $i \in S_k$. Hence, if it is possible to cheaply compute these gradients *without* actually forming y_k, we can avoid full-dimensional operations.

We now show that this can be done for functions f of the form (2), where f_j is as in Theorem 2:

$$f(x) = \sum_{j=1}^{m} \phi_j(e_j^T A x). \tag{63}$$

Let D_i be the set of such j for which $A_{ji} \neq 0$. If we write $r_{u_k} = Au_k$ and $r_{\tilde{z}_k} = A\tilde{z}_k$, then, using (63), we can write

$$\nabla_i f(\theta_k^2 u_k + \tilde{z}_k) = \sum_{j \in D_i} A_{ji}^T \phi_j'(\theta_k^2 r_{u_k}^j + r_{\tilde{z}_k}^j). \tag{64}$$

Assuming we store and maintain the residuals r_{u_k} and $r_{\tilde{z}_k}$, the computation of the product $A_{ji}^T \phi_j'(\cdot)$ costs $\mathcal{O}(N_i)$ (we assume that the evaluation of the univariate derivative ϕ_j' takes $\mathcal{O}(1)$ time), and hence the computation of the block derivative (64) requires $\mathcal{O}(|D_i|N_i)$ arithmetic operations. Hence on average, computing all block gradients for $i \in S_k$ will cost

$$C = \mathbf{E}\left[\sum_{i \in \hat{S}} \mathcal{O}(|D_i|N_i)\right] = \frac{\tau}{n} \sum_{i=1}^{n} \mathcal{O}(|D_i|N_i).$$

This will be small if $|D_i|$ are small and τ is small. For simplicity, assume all blocks are of equal size, $N_i = b = N/n$. Then

$$C = \frac{b\tau}{n} \times \mathcal{O}\left(\sum_{i=1}^{n} |D_i|\right) = \frac{b\tau}{n} \times \mathcal{O}\left(\sum_{j=1}^{m} \omega_j\right) = \frac{b\tau m}{n} \mathcal{O}(\tilde{\omega}) = \tau \times \mathcal{O}\left(\frac{bm\tilde{\omega}}{n}\right),$$

where $\tilde{\omega} = \frac{1}{m}\sum_j \omega_j$. It can be easily shown that the maintenance of the residual vectors r_{u_k} and $r_{\tilde{z}_k}$ takes the same amount of time (C), and hence the total work per iteration is C. In many practical situations, $m \leq n$, and often $m \ll n$ (we focus on this case in the paper since usually this corresponds to f not being strongly convex) and $\bar{\omega} = O(1)$. This then means that $C = \tau \times \mathcal{O}(b)$. That is, each of the τ processors do work proportional to the size of a single block per iteration.

The favorable situation described above is the consequence of the block sparsity of the data matrix A and does not depend on ϕ_j insofar as the evaluation of its derivative takes $\mathcal{O}(1)$ work. Hence, it applies to convex quadratics ($\phi_j(s) = s^2$), to logistic regression ($\phi_j(r) = \log(1 + \exp(s))$), and also to the smooth approximation $f_\mu(x)$ of $f(x) = \|Ax - b\|_1$, defined by

$$f_\mu(x) = \sum_{j=1}^{m} \|e_j^T A\|_{w^*}^* \psi_\mu\left(\frac{|e_j^T Ax - b_j|}{\|e_j^T A\|_v^*}\right), \qquad \psi_\mu(t) = \begin{cases} \frac{t^2}{2\mu}, & 0 \le t \le \mu, \\ t - \frac{\mu}{2}, & \mu \le t, \end{cases}$$

with smoothing parameter $\mu > 0$, as considered in [22, 11]. Vector w^* is as defined in [11]; $\|\cdot\|_v$ is a weighted norm in $\mathbf{R}^m$.

7. Numerical Experiments. In all tests we used a shared-memory workstation with 4 Intel Xeon X5670 processors (24 cores in total) at 2.93 GHz and 192 GB RAM. In the experiments, we have departed from the theory in two ways:

(i) Our implementation of APPROX is *asynchronous* in order to limit communication costs. For example, on the problem of section 7.2, the asynchronous implementation is about 5 times faster than the synchronous implementation, where each processor waits until the others terminate their update of the variable before proceeding.

(ii) We approximated the τ-nice sampling by a τ-independent sampling as in [33] (the latter is very easy to generate in parallel; please note that our analysis can be very easily extended to cover the τ-independent sampling).

For simplicity, in all tests we assume all blocks are of size 1 ($N_i = 1$ for all i). However, further speedups can be obtained by working with larger block sizes as then each processor is better utilized. For the problems we consider, coordinate descent methods are the state of the art. Hence, all methods we compare are coordinate descent methods of some variety. These methods share many similar components (e.g., computation of partial derivative, update of a coordinate, sampling); wherever possible, in our comparisons we have built all methods using identical components. Hence, while we often report runtime in the experiments, all differences are a genuine reflection of differences of the algorithms.

7.1. The Effect of New Stepsizes. In this experiment, we compare the performance of the new stepsizes (introduced in section 3.2) with those proposed in [33] (see Table 5). We generated random LASSO (L_1-regularized least-squares) instances

$$f(x) = \frac{1}{2}\|Ax - b\|^2, \qquad \psi(x) = \lambda\|x\|_1,$$

with various distributions of the separability degrees ω_j (= number of nonzero elements on the jth row of A) and studied the weighted distance to the optimum $\|x_* - x_0\|_v$ for the initial point $x_0 = 0$. This quantity appears in the complexity estimate (37) and depends on τ (the number of processors). We chose a random matrix of small size, $N = m = 1000$, as this is sufficient to make our point, and consider $\tau \in \{10, 100, 1000\}$. In particular, we consider three different distributions of $\{\omega_j\}$: uniform, intermediate, and extreme. The results are summarized in Table 7. First, we generated a *uniformly* sparse matrix with $\omega_j = 30$ for all j. In this case, $v^{\rm fr} = v^{\rm rt}$, and hence the results are the same. We then generated an *intermediate* instance, with $\omega_j = 1 + \lfloor 30j^2/m^2 \rfloor$. The matrix has many rows with a few nonzero elements and some rows with up to 30 nonzero elements. Looking at the table, clearly, the new

Table 7 *Comparison of ESOs in the uniform case.*

	Uniform		Intermediate		Extreme	
τ	$\|x^*\|_{v^{\rm fr}}$	$\|x^*\|_{v^{\rm rt}}$	$\|x^*\|_{v^{\rm fr}}$	$\|x^*\|_{v^{\rm rt}}$	$\|x^*\|_{v^{\rm fr}}$	$\|x^*\|_{v^{\rm rt}}$
10	10.82	10.82	6.12	6.43	2.78	5.43
100	19.00	19.00	9.30	11.38	4.31	16.08
1000	52.49	52.49	24.00	31.78	11.32	50.52

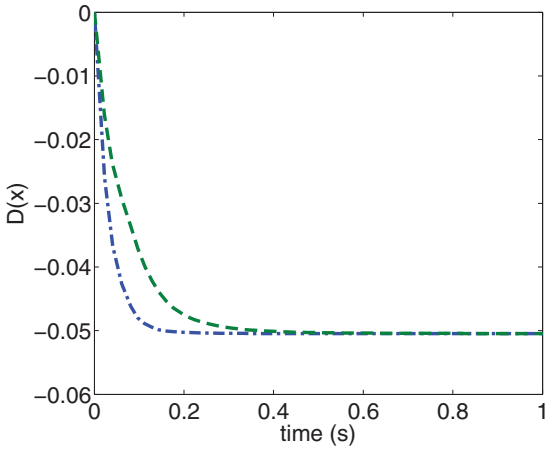

Fig. 1 *Comparison of new (dash-dotted line) and old (dashed line) stepsizes for the dual SVM problem on the* `rcv1` *dataset. We used only $\tau = 8$ processors and the new stepsizes already lead to a two times speedup.*

stepsizes are better. The improvement is moderate when there are a few processors, but for $\tau = 1000$, the complexity is 25% better. Finally, we generated a rather *extreme* matrix with $\omega_1 = 500$ and $\omega_j = 3$ for $j > 1$. We can see that the new stepsizes are much better, even with few processors, and can lead to a $5\times$ speedup.

In the experiments above, we have first fixed a sparsity pattern and then generated a *random* matrix A based on it. However, much larger differences can be seen for special matrices A. Consider the case $\tau = n$. In view of (39), the complexity of APPROX is proportional to $\|v\|_1$. Fix ω and $\omega_1, \ldots, \omega_j$ and let us ask the question, for what data matrix A will the ratio $\theta = \|v^{\rm rt}\|_1/\|v^{\rm fr}\|_1$ be maximized? Since $\|v^{\rm rt}\|_1 = \omega \sum_j \|A_{j:}\|^2$ and $\|v^{\rm fr}\|_1 = \sum_j \omega_j \|A_{j:}\|^2$, the maximal ratio is given by

$$\max_A \theta \stackrel{\rm def}{=} \max_{\alpha \geq 0} \left\{ \omega \sum_{j=1}^m \alpha_j \ : \ \sum_{j=1}^m \omega_j \alpha_j \leq 1 \right\} = \max_j \frac{\omega}{\omega_j}.$$

The extreme case is attained for some matrix with at least one dense row (ω_j) and one maximally sparse row ($\omega_j = 1$), leading to $\theta = n$. So, there are instances for which the new stepsizes can lead to an up to $n\times$ speedup for APPROX when compared to the stepsizes $v^{\rm rt}$. Needless to say, these extreme instances are artificially constructed.

In Figure 1, we give the value of the SVM dual objective when minimized by serial randomized coordinate descent [32] (see section 7.4 for details on this problem). A similar plot would be obtained with APPROX. The dataset is `rcv1` [24]. It consists of a matrix A with $m = 47{,}236$ and $N = 20{,}242$ and a vector b. The new stepsizes

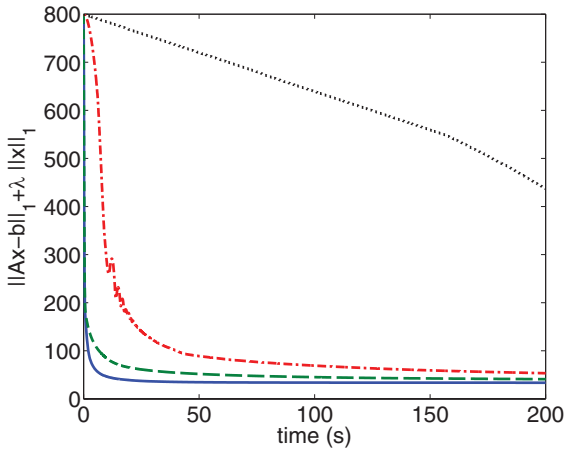

Fig. 2 *Comparison of four algorithms for L_1-regularized L_1 regression on the* `dorothea` *dataset: gradient method (dotted black line), accelerated gradient method ([22], dash-dotted red line), smoothed parallel coordinate descent method (SPCDM [11], dashed green line) with stepsizes v^{fr} and APPROX with stepsizes v^{fr} (solid blue line).*

are very useful for this problem since $\omega = \max_j \omega_j = 8{,}551$ and $\bar{\omega} \approx 488 < \omega/17$ (see Theorem 1 for the definition of $\bar{\omega}$). In all subsequent experiments we consider these new stepsizes, be it for accelerated or nonaccelerated parallel coordinate descent.

7.2. L1-Regularized L1 Regression. We wish to find $x \in \mathbb{R}^N$ that minimizes

$$\|Ax - b\|_1 + \lambda \|x\|_1$$

with $\lambda = 1$. Because the objective is nonsmooth and nonseparable, we apply the smoothing technique presented in [22] for the first part of the objective and use the smoothed parallel coordinate descent method (SPCDM) proposed in [11]. The level of smoothing depends on the expected accuracy: we chose $\epsilon = 0.1$ (0.0125% of the initial value obtained at $x_0 = 0$), so the smoothing parameter defined in [11] is $\mu = 4.2 \times 10^{-6}$.

We consider the `dorothea` dataset [24]. It is a sparse moderate-sized feature matrix A with $m = 800$, $N = 100{,}000$, $\omega = 6{,}061$, $\bar{\omega} \approx 1{,}104.1$, and a vector $b \in \mathbf{R}^m$.

We compared 4 algorithms (see Figure 2), all run with 4 processors ($\tau = 4$). As one can see, coordinate descent methods are much more efficient on this problem than both gradient descent and accelerated gradient descent. Also, APPROX is better than SPCDM. As the problem is of small size, we could compute the optimal solution using an interior point method for linear programming and compare the value at each iteration to the optimal value (Table 8). Each line of the table gives the time needed by APPROX and PCDM to reach a given accuracy target. In the beginning (until $F(x_k) - F(x^*) < 6.4$), the algorithms are in a transitional phase. Then, when one runs the algorithm twice as long, $F(x_k) - F(x^*)$ is divided by 2 for SPCDM and by 4 for APPROX. This highlights the difference in the convergence speeds: $O(1/k)$ compared to $O(1/k^2)$. As a result, APPROX gives an ϵ-solution in 186.5 seconds, while SPCDM has not finished yet after 2,000 seconds.

7.3. LASSO. We now consider L_1-regularized least-squares regression on the `KDDB` dataset [42]. It consists of a large sparse matrix $A \in \mathbf{R}^{m \times N}$ with $m = 29{,}890{,}095$, $N = 19{,}264{,}097$ (with $\omega = 75$ and $\bar{\omega} \approx 31.87$), and a vector $b \in \mathbf{R}^m$. As is standard

Table 8 *Comparison of objective decreases for APPROX and smoothed parallel coordinate descent (SPCDM) on a problem with* $F(x) = \|Ax - b\|_1 + \lambda\|x\|_1$.

$F(x_k) - F(x_*)$	**APPROX**	SPCDM [11]
409.6	0.3 s	0.3 s
204.8	0.4 s	0.5 s
102.4	1.3 s	2.8 s
51.2	2.7 s	9.9 s
25.6	5.5 s	28.6 s
12.8	10.1 s	87.0 s
6.4	17.8 s	252.4 s
3.2	27.7 s	526.3 s
1.6	41.2 s	1,041.1 s
0.8	58.7 s	1,896.5 s
0.4	86.8 s	>2,000 s
0.2	122.7 s	>2,000 s
0.1	186.5 s	>2,000 s

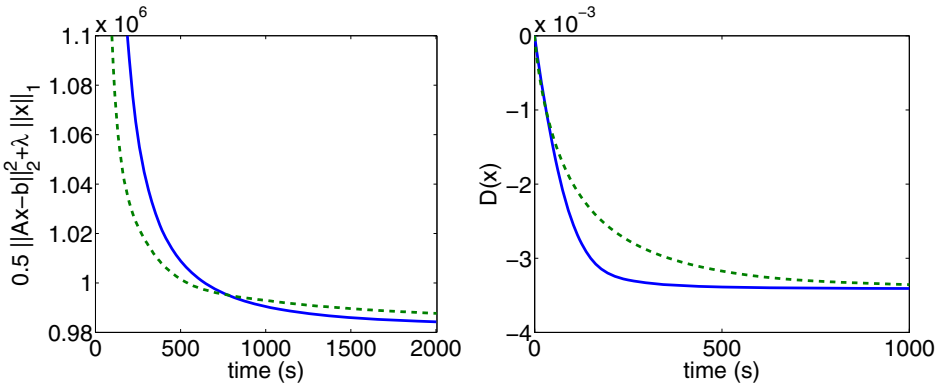

Fig. 3 *Comparison of PCDM ([33]; dashed line) and APPROX (solid line).* Left: l_1*-regularized least-squares problem and the* `KDDB` *dataset with* $\lambda = \lambda_{\max}/10^3$. *As the decrease is big in the beginning (from* 8.3×10^6 *to* 1.1×10^6*), we present a zoom for* $9.8 \times 10^5 \leq F(x) \leq 1.1 \times 10^6$. Right: *SVM dual problem and the* `Malicious URL` *dataset.*

practice for the LASSO problem, we normalized the columns of the matrix A. We wish to find $x \in \mathbf{R}^N$ that minimizes

$$F(x) = \frac{1}{2}\|Ax - b\|^2 + \lambda\|x\|_1.$$

We compare APPROX (Algorithm 2) with the (nonaccelerated) parallel coordinate descent method (PCDM [33]) in Figure 3 (left), both run with $\tau = 16$ processors and $\lambda = \lambda_{\max}/10^3$, where $\lambda_{\max} = \max_{1 \leq j \leq m} |(A^T b)_j|$ is the smallest regularization parameter for which 0 is solution to the LASSO problem. Coordinate descent is currently the method of choice of many solvers for the LASSO problem [50]. Both algorithms converge quickly. PCDM is faster in the beginning because each iteration is half as expensive. However, APPROX is faster afterwards. In Table 9 we give the accuracy achieved by PCDM and APPROX for a wide range of regularization parameters. We obtained feasible dual points by dual scaling of the residuals [8]. We can see that, except for the highly regularized problem, APPROX indeed is faster.

Table 9 *Comparison of LASSO problem's duality gap after* 2,000 *s of computations for APPROX and smoothed parallel coordinate descent (SPCDM) on the* KDDB *dataset. We give the value as a percentage of the initial duality gap obtained at* $x_0 = 0$. *We obtained a sparse solution with the postprocessing described in the last paragraph of this section.*

λ	$\lambda_{\max}/10$	$\lambda_{\max}/10^2$	$\lambda_{\max}/10^3$	$\lambda_{\max}/10^4$	$\lambda_{\max}/10^5$
$\mathrm{nnz}(x_*) \approx$	58	570	96,000	3.7×10^6	8.1×10^6
APPROX	0.017%	0.025%	0.100%	1.890%	0.543%
PCDM [33]	0%	0.299%	3.794%	3.383%	0.669%

Table 10 *Decrease of the duality gap for APPROX and stochastic dual coordinate ascent (SDCA).*

Duality gap	**APPROX**	SDCA
0.032	24 s	30 s
0.016	56 s	95 s
0.008	104 s	225 s
0.004	144 s	390 s
0.002	216 s	603 s
0.001	290 s	936 s
0.0005	420 s	1,359 s
0.00025	616 s	1,951 s

An important feature of the L_1-regularization is that it promotes sparsity in the optimization variable x. As APPROX only involves proximal steps on the z variable, only z_k is encouraged to be sparse, and not x_k, y_k, or u_k. A possible way to obtain a sparse solution is to first compute x_k and then postprocess with a few iterations of a sparsity-oriented method (such as iterative hard thresholding, proximal gradient descent, or cyclic/randomized coordinate descent).

7.4. Training Linear Support Vector Machines. Our last experiment is the dual of the SVM problem [38]. For the dual SVM, the coordinates correspond to examples. We use the `Malicious URL` dataset [24] with data matrix A of size $m = 3{,}231{,}961$, $N = 2{,}396{,}130$, and a vector $b \in \mathbb{R}^N$. We adapted b so that $b_i \in \{-1, 1\}$. Here $\omega = N = n$ but the matrix is still sparse: $nnz(A) = 277{,}058{,}644 \ll mn$, $\frac{1}{m}\sum_{j=1}^{m} \omega_j \approx 85.7$. Our goal is to find $x \in [0,1]^N$ that minimizes

$$D(x) = \frac{1}{2\lambda N^2} \sum_{j=1}^{m} \left(\sum_{i=1}^{N} b_i A_{ji} x_i \right)^2 - \frac{1}{N} \sum_{i=1}^{N} x_i + I_{[0,1]^N}(x),$$

with $\lambda = 1/N$. We compare APPROX (Algorithm 2) with stochastic dual coordinate ascent (SDCA [32, 40, 44]); the results are in Figure 3 (right). We have used a single processor only ($\tau = 1$) because $\omega = N$ and $\bar{\omega} \approx 1.6 \times 10^6 \approx 2N/3$ are quite large, making parallelization not so efficient. For this problem, one can recover the primal solution [38], and thus we can compare the decrease in the duality gap, summarized in Table 10. APPROX is two to three times as fast as SDCA on this instance.

An alternative algorithm is accelerated SDCA [41]. But to apply it, one needs to smooth the $L1$ norm, which makes the problem ill conditioned. In a similar fashion to LASSO (see Table 6), the complexity bound of accelerated SDCA involves logarithmic terms of order $\log(\frac{vN}{\epsilon})^2 \approx 200$ that are not negligible in the present case and make this algorithm not competitive when compared with SDCA or APPROX.

8. Conclusion. In summary, we proposed APPROX, a randomized coordinate descent method combining the following *four acceleration strategies:*

1. Our method is *accelerated*, i.e., it achieves $O(1/k^2)$ convergence rate (in expectation). Hence, the method is better able to obtain a high-accuracy solution on nonstrongly convex problem instances than nonaccelerated methods, which achieve the slower rate $O(1/k)$.
2. Our method is *parallel*. Hence, it is able to better utilize modern parallel computing architectures and effectively tame the problem dimension n.
3. We proposed new *longer stepsizes* for faster convergence on functions whose degree of separability ω is larger than their average degree of separability $\bar{\omega}$.
4. We have shown that our method can be implemented *without the need to perform full-dimensional vector operations.*

Our work is amenable to further extensions. When the function to minimize is strongly convex and its coefficient of strong convexity is known, one can design a specialized algorithm (see the follow-up papers [25, 15]). Further, in this paper we have focused on the case of *uniform* samplings. However, with a proper change in the definition of ESO, one can also handle *nonuniform* samplings [29, 28, 27] as well. A particular type of nonuniform (and, in fact, nonstationary) sampling is the shrinking technique, which is often used when solving the LASSO problem with classical coordinate descent [32]. The idea is to update more often the nonzero coordinates than the coordinates that we think are zeros at the optimum. The main issues when combining this idea with APPROX are (i) the algorithm depends explicitly on the number of variables, and (ii) even if $z_k^{(i)} = 0$ for all $k \geq k_0$, we may have $x_k^{(i)} \neq 0$. It may be necessary to restart the algorithm from time to time in order to overcome these issues. Finally, it would be interesting to analyze APPROX used with an adaptive sampling strategy, such as the one employed in [6].

Appendix A. Proof of Proposition 1 (equivalence). It is straightforward to see that $x_0 = y_0 = z_0 = \tilde{x}_0 = \tilde{y}_0 = \tilde{z}_0$, and hence the statement holds for $k = 0$. By induction, assume it holds for some k. Note that for $i \notin S_k$, $\tilde{z}_{k+1}^{(i)} = \tilde{z}_k^{(i)} = z_k^{(i)} = z_{k+1}^{(i)}$. If $i \in S_k$, then

$$\tilde{z}_{k+1}^{(i)} = \tilde{z}_k^{(i)} + t_k^{(i)}, \tag{65}$$

where

$$\begin{aligned}
t_k^{(i)} &= \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(\theta_k^2 u_k + \tilde{z}_k), t \rangle + \frac{n\theta_k v_i}{2\tau} \|t\|_{(i)}^2 + \psi_i(\tilde{z}_k^{(i)} + t) \right\} \\
&\overset{(62)}{=} \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(\tilde{y}_k), t \rangle + \frac{n\theta_k v_i}{2\tau} \|t\|_{(i)}^2 + \psi_i(\tilde{z}_k^{(i)} + t) \right\} \\
&= \arg\min_{t \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(y_k), t \rangle + \frac{n\theta_k v_i}{2\tau} \|t\|_{(i)}^2 + \psi_i(z_k^{(i)} + t) \right\} \\
&= -z_k^{(i)} + \arg\min_{z \in \mathbf{R}^{N_i}} \left\{ \langle \nabla_i f(y_k), z - y_k^{(i)} \rangle + \frac{n\theta_k v_i}{2\tau} \|z - z_k^{(i)}\|_{(i)}^2 + \psi_i(z) \right\} \\
&= -z_k^{(i)} + z_{k+1}^{(i)}. \qquad (66)
\end{aligned}$$

Combining (65) with (66), we get $\tilde{z}_{k+1}^{(i)} = \tilde{z}_k^{(i)} - z_k^{(i)} + z_{k+1}^{(i)} = z_{k+1}^{(i)}$. Further, combining the two cases ($i \in S_k$ and $i \notin S_k$), we arrive at

$$\tilde{z}_{k+1} = z_{k+1}. \tag{67}$$

Looking at Algorithm 2, we see that $u_{k+1} - u_k = -\frac{1-\frac{n}{\tau}\theta_k}{\theta_k^2}(\tilde{z}_{k+1} - \tilde{z}_k)$, and thus

$$\begin{aligned}\tilde{x}_{k+1} &\overset{(61)}{=} \theta_k^2 u_{k+1} + \tilde{z}_{k+1} = \theta_k^2\left(u_k - \frac{1-\frac{n}{\tau}\theta_k}{\theta_k^2}(\tilde{z}_{k+1} - \tilde{z}_k)\right) + \tilde{z}_{k+1} \\ &= \theta_k^2 u_k + \tilde{z}_k + \frac{n}{\tau}\theta_k(\tilde{z}_{k+1} - \tilde{z}_k) \\ &\overset{(62)}{=} \tilde{y}_k + \frac{n}{\tau}\theta_k(\tilde{z}_{k+1} - \tilde{z}_k) \overset{(67)}{=} y_k + \frac{n}{\tau}\theta_k(z_{k+1} - z_k) = x_{k+1}. \end{aligned} \tag{68}$$

Finally,

$$\begin{aligned}\tilde{y}_{k+1} &\overset{(62)}{=} \theta_{k+1}^2 u_{k+1} + \tilde{z}_{k+1} \overset{(61)}{=} \frac{\theta_{k+1}^2}{\theta_k^2}(\tilde{x}_{k+1} - \tilde{z}_{k+1}) + \tilde{z}_{k+1} \\ &\overset{(43)}{=} (1-\theta_{k+1})(\tilde{x}_{k+1} - \tilde{z}_{k+1}) + \tilde{z}_{k+1} \overset{(67)+(68)}{=} (1-\theta_{k+1})(x_{k+1} - z_{k+1}) + z_{k+1} \\ &= y_{k+1}. \end{aligned}$$

REFERENCES

[1] Z. Allen-Zhu and L. Orecchia, *Nearly-linear time packing and covering LP solver with faster convergence rate than $O(1/\varepsilon^2)$*, in Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing (STOC 2015), 2015. (Cited on pp. 6, 9)

[2] F. Bach, *Learning with submodular functions: A convex optimization perspective*, Found. Trends Mach. Learning, 6 (2013), pp. 145–373. (Cited on p. 8)

[3] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM J. Imaging Sci., 2 (2009), pp. 183–202, doi:10.1137/080716542. (Cited on pp. 2, 4)

[4] A. Ben-Tal and A. Nemirovski, *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, MOS-SIAM Ser. Optim. 2, SIAM, Philadelphia, 2001. (Cited on p. 9)

[5] J. K. Bradley, A. Kyrola, D. Bickson, and C. Guestrin, *Parallel coordinate descent for L1-regularized loss minimization*, in Proceedings of the 28th International Conference on Machine Learning, 2011. (Cited on pp. 2, 3)

[6] D. Csiba, Z. Qu, and P. Richtárik, *Stochastic dual coordinate ascent with adaptive probabilities*, in Proceedings of the 32nd International Conference on Machine Learning (ICML), Lille, France, 2015. (Cited on p. 30)

[7] D. Csiba and P. Richtárik, *Coordinate Descent Face-Off: Primal or Dual?*, preprint, arXiv:1605.08982v1 [math.OC], 2016. (Cited on p. 7)

[8] L. El Ghaoui, V. Viallon, and T. Rabbani, *Safe feature elimination in sparse supervised learning*, Pacific J. Optim., 8 (2012), pp. 667–698. (Cited on p. 28)

[9] A. Ene and H. L. Nguyen, *Random coordinate descent methods for minimizing decomposable submodular functions*, in Proceedings of the 32nd International Conference on Machine Learning (ICML), Lille, France, 2015 (Cited on pp. 6, 8)

[10] O. Fercoq, Z. Qu, P. Richtárik, and M. Takáč, *Fast distributed coordinate descent for minimizing non-strongly convex losses*, in IEEE International Workshop on Machine Learning for Signal Processing, 2014. (Cited on pp. 4, 6, 7, 10)

[11] O. Fercoq and P. Richtárik, *Smooth Minimization of Nonsmooth Functions by Parallel Coordinate Descent*, preprint, arXiv:1309.5885v1 [cs.DC], 2013. (Cited on pp. 2, 3, 4, 10, 11, 25, 27, 28)

[12] O. Fercoq and P. Richtárik, *Accelerated, parallel, and proximal coordinate descent*, SIAM J. Optim., 25 (2015), pp. 1997–2023, doi:10.1137/130949993. (Cited on pp. 4, 6, 10, 12)

[13] Y. T. Lee and A. Sidford, *Efficient accelerated coordinate descent methods and faster algorithms for solving linear systems*, in Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2013), 2013. (Cited on pp. 2, 3, 23)

[14] D. Leventhal and A. Lewis, *Randomized methods for linear constraints: Convergence rates and conditioning*, Math. Oper. Res., 35 (2010), pp. 641–654. (Cited on p. 3)

[15] Q. Lin, Z. Lu, and L. Xiao, *An accelerated randomized proximal coordinate gradient method and its application to regularized empirical risk minimization*, SIAM J. Optim., 25 (2015), pp. 2244–2273, doi:10.1137/141000270. (Cited on pp. 6, 8, 30)

[16] J. Liu, S. J. Wright, C. Ré, V. Bittorf, and S. Sridhar, *An asynchronous parallel stochastic coordinate descent algorithm*, J. Mach. Learn. Res., 16 (2015), pp. 285–322. (Cited on p. 3)

[17] J. Mareček, P. Richtárik, and M. Takáč, *Distributed block coordinate descent for minimizing partially separable functions*, in Numerical Analysis and Optimization, Springer Proc. Math. Stat. 134, Springer, Cham, 2015, pp. 261–288. (Cited on pp. 2, 7)

[18] I. Necoara and D. Clipici, *Efficient parallel coordinate descent algorithm for convex optimization problems with separable constraints: Application to distributed MPC*, J. Process Control, 23 (2013), pp. 243–253. (Cited on pp. 2, 3)

[19] I. Necoara and D. Clipici, *Parallel random coordinate descent method for composite minimization: Convergence analysis and error bounds*, SIAM J. Optim., 26 (2016), pp. 197–226, doi:10.1137/130950288. (Cited on pp. 2, 15)

[20] I. Necoara, Yu. Nesterov, and F. Glineur, *Efficiency of Randomized Coordinate Descent Methods on Optimization Problems with Linearly Coupled Constraints*, tech. report, Politehnica University of Bucharest, 2012. (Cited on p. 3)

[21] Yu. Nesterov, *A method of solving a convex programming problem with convergence rate O(1/k2)*, Soviet Math. Dokl., 27 (1983), pp. 372–376. (Cited on p. 2)

[22] Yu. Nesterov, *Smooth minimization of nonsmooth functions*, Math. Programming, 103 (2005), pp. 127–152. (Cited on pp. 25, 27)

[23] Yu. Nesterov, *Efficiency of coordinate descent methods on huge-scale optimization problems*, SIAM J. Optim., 22 (2012), pp. 341–362, doi:10.1137/100802001. (Cited on pp. 2, 3, 18)

[24] J. C. Platt, *Fast training of support vector machines using sequential minimal optimization*, in Advances in Kernel Methods: Support Vector Learning, B. Scholkopf, C. Burges, and A. Smola, eds., MIT Press, Cambridge, MA, 1999, pp. 185–208, http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets. (Cited on pp. 26, 27, 29)

[25] Z. Qu, O. Fercoq, and P. Richtárik, *Accelerated Coordinate Descent Method for Strongly Convex Functions*, tech. report 05/2014. (Cited on p. 30)

[26] Z. Qu and P. Richtárik, *Coordinate descent with arbitrary sampling* I: *Algorithms and complexity*, Optim. Methods Softw., (2016), doi:10.1080/10556788.2016.1190360. (Cited on pp. 2, 10)

[27] Z. Qu and P. Richtárik, *Coordinate descent with arbitrary sampling* II: *Expected separable overapproximation*, Optim. Methods Softw., (2016). (Cited on pp. 12, 30)

[28] Z. Qu, P. Richtárik, and T. Zhang, *Quartz: Randomized dual coordinate ascent with arbitrary sampling*, in Advances in Neural Information Processing Systems 28 (NIPS 2015), 2015, pp. 865–873. (Cited on p. 30)

[29] P. Richtárik and M. Takáč, *On optimal probabilities in stochastic coordinate descent methods*, Optim. Lett., (2015), pp. 1–11. (Cited on pp. 2, 3, 10, 30)

[30] P. Richtárik and M. Takáč, *Distributed coordinate descent method for learning with big data*, J. Mach. Learn. Res., 17 (2016), pp. 1–25. (Cited on pp. 2, 3, 4, 10)

[31] P. Richtárik and M. Takáč, *Distributed coordinate descent method for learning with big data*, 17 (2016), pp. 1–25. (Cited on p. 7)

[32] P. Richtárik and M. Takáč, *Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function*, Math. Program., 144 (2014), pp. 1–38. (Cited on pp. 2, 3, 4, 7, 18, 19, 26, 29, 30)

[33] P. Richtárik and M. Takáč, *Parallel coordinate descent methods for big data optimization problems*, Math. Program., 156 (2016), pp. 433–484. (Cited on pp. 2, 3, 4, 7, 10, 11, 12, 13, 14, 15, 17, 21, 23, 25, 28, 29)

[34] R. T. Rockafellar, *Augmented Lagrangians and applications of the proximal point algorithm in convex programming*, Math. Oper. Res., 1 (1976), pp. 97–116. (Cited on p. 9)

[35] R. T. Rockafellar, *Monotone operators and the proximal point algorithm*, SIAM J. Control Optim., 14 (1976), pp. 877–898, doi:10.1137/0314056. (Cited on p. 9)

[36] C. Rother, V. Kolmogorov, and A. Blake, *GrabCut: Interactive foreground extraction using iterated graph cuts*, ACM Trans. Graphics, 23 (2004), pp. 309–314. (Cited on p. 8)

[37] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, New York, 2014. (Cited on p. 6)

[38] S. Shalev-Shwartz and A. Tewari, *Stochastic methods for ℓ_1-regularized loss minimization*, J. Mach. Learn. Res., 12 (2011), pp. 1865–1892. (Cited on pp. 3, 18, 29)

[39] S. Shalev-Shwartz and T. Zhang, *Proximal Stochastic Dual Coordinate Ascent*, preprint, arXiv:1211.2717v1 [stat.ML], 2012. (Cited on p. 3)

[40] S. Shalev-Shwartz and T. Zhang, *Stochastic dual coordinate ascent methods for regularized loss minimization*, J. Mach. Learn. Res., 14 (2013), pp. 567–599. (Cited on pp. 7, 29)

[41] S. Shalev-Shwartz and T. Zhang, *Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization*, Math. Program., 155 (2016), pp. 105–145. (Cited on pp. 3, 19, 29)

[42] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. J. Gordon, and K. R. Koedinger, *Bridge to algebra* 2008–2009. *Challenge data set from KDD Cup* 2010 *Educational Data Mining Challenge*, 2010. (Cited on p. 27)

[43] D. Sun, K.-C. Toh, and L. Yang, *An efficient inexact ABCD method for least squares semidefinite programming*, SIAM J. Optim., 26 (2016), pp. 1072–1100, doi:10.1137/15M1021799. (Cited on pp. 6, 9)

[44] M. Takáč, A. Bijral, P. Richtárik, and N. Srebro, *Mini-batch primal and dual methods for SVMs*, in Proceedings of the 30th International Conference on Machine Learning, 2013. (Cited on pp. 2, 3, 29)

[45] R. Tappenden, P. Richtárik, and J. Gondzio, *Inexact block coordinate descent method: Complexity and preconditioning*, J. Optim. Theory Appl., 170 (2016), pp. 144–176. (Cited on p. 3)

[46] M. J. Todd, *Semidefinite optimization*, Acta Numer., 10 (2001), pp. 515–560. (Cited on p. 9)

[47] P. Tseng, *On accelerated proximal gradient methods for convex-concave optimization*, Submitted to SIAM J. Optim., (2008). (Cited on pp. 2, 4, 17, 19, 20)

[48] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Rev., 38 (1996), pp. 49–95, doi:10.1137/1038003. (Cited on p. 9)

[49] H. Wolkowicz, R. Saigal, and L. Vandenberghe, eds., *Handbook of Semidefinite Programming*, Internat. Ser. Oper. Res. Management Sci. 27, Springer, New York, 2010. (Cited on p. 9)

[50] T. T. Wu and K. Lange, *Coordinate descent algorithms for lasso penalized regression*, Ann. Appl. Stat., 2 (2008), pp. 224–244. (Cited on p. 28)

[51] L. Xiao and Z. Lu, *On the complexity analysis of randomized block-coordinate descent methods*, Math. Program., 152 (2015), pp. 615–642. (Cited on p. 3)